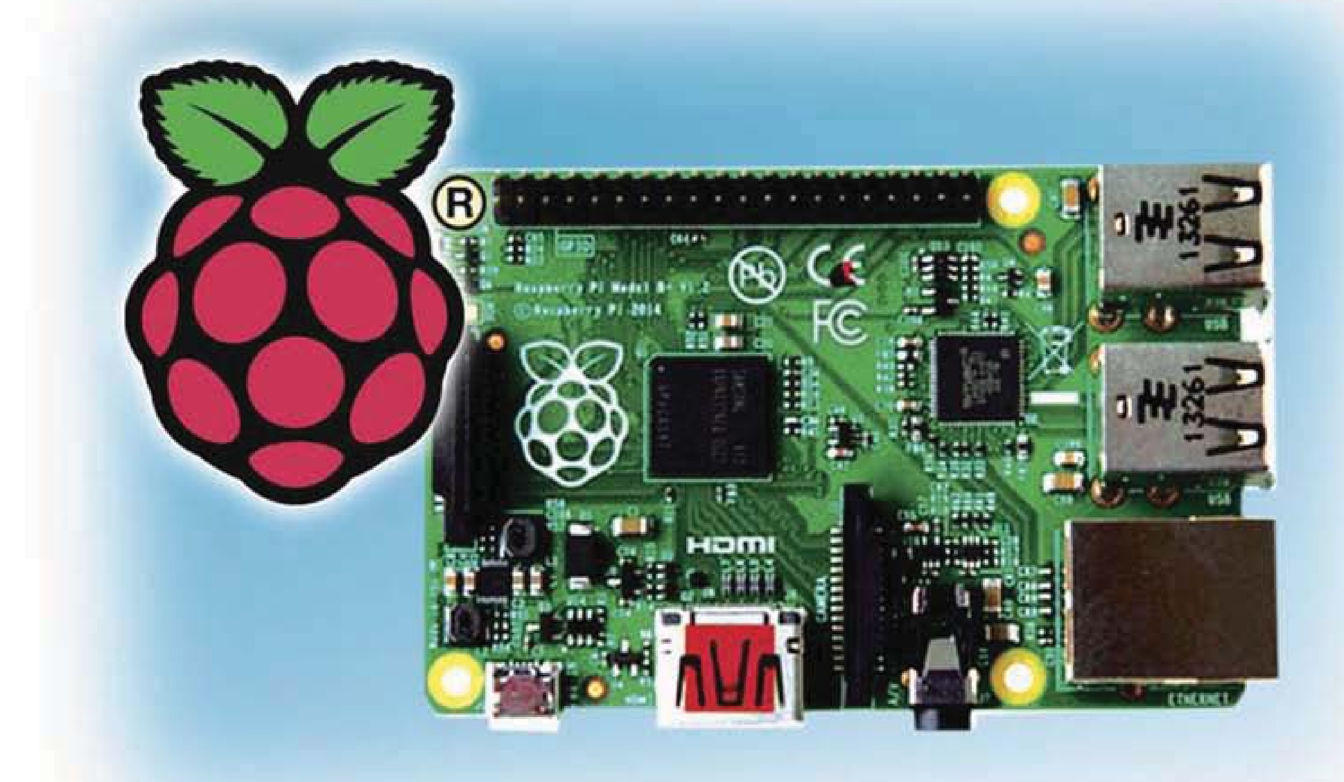
Электроника

В. А. Петин





Микрокомпьютеры Raspberry Pi

Практическое руководство



- Подбор и настройка оборудования
- Операционные системы для Raspberry Pi
- Проекты использования Raspberry Pi
- Медиаплеер RaspBMC
- Контакты GPIO и платы расширения

Raspberry Pi — мини-компьютер с макси-возможностями

Виктор Петин

Микрокомпьютеры Raspberry Pi

Практическое руководство

Санкт-Петербург
«БХВ-Петербург»
2015

УДК 004
ББК 32.973
П29

Петин В. А.

П29 Микрокомпьютеры Raspberry Pi. Практическое руководство. — СПб.: БХВ-Петербург, 2015. — 240 с.: ил. — (Электроника)

ISBN 978-5-9775-3519-9

Рассмотрены вопросы подбора и настройки периферийных устройств для микрокомпьютеров Raspberry Pi. Подробно описана установка операционной системы. Большая часть материала посвящена работе с дистрибутивом Raspbian. Описаны настройка и установка дополнительных пакетов, удаленный доступ к компьютеру с помощью SSH и VNC, использование Raspberry Pi в качестве веб-сервера, torrent-клиента, сервера видеонаблюдения, голосовое управление компьютером, взаимодействие с библиотекой "компьютерного зрения" openCV, операционной системой роботов ROS, платой Arduino и многое другое. Рассмотрено применение Raspberry Pi в качестве медиаплеера XBMC. Разобрано использование выводов GPIO и платы расширения Gertboard и XMOS Starter Kit для Raspberry.

На сайте издательства размещен архив с примерами и проектами из книги.

Для разработчиков

УДК 004
ББК 32.973

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 29.08.14.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 19,35.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3519-9

Оглавление



ГЛАВА 1

Общие сведения

Raspberry Pi — это маленький, размером с кредитную карту, компьютер стоимостью порядка 25 долларов за базовую модель и 35 — за более продвинутую. Raspberry Pi основан на процессоре с архитектурой ARM11 и частотой в 700 МГц. В последних версиях прошивки официально разрешили разгонять процессор до 1000 МГц, что обеспечивает достижение приемлемой производительности при низком энергопотреблении. Raspberry Pi по сути представляет собой полноценный системный блок, с помощью которого можно обучать работе с компьютером, воспроизводить видео, программировать, пользоваться Интернетом, слушать музыку. Одна из главных и привлекательных особенностей Raspberry Pi — наличие на плате аппаратных портов ввода/вывода GPIO (General Purpose Input/Output, входы/выходы общего назначения), что открывает перспективы использования его в робототехнических проектах.

1.1. История создания

История появления и развития Raspberry Pi берет начало с 2006 года. Основатели проекта — сотрудники и преподаватели Компьютерной лаборатории Кембриджского университета. Одной из задач инициативы явилось возрождение интереса подрастающего поколения к программированию на достаточно продвинутом уровне. Предполагалось, что основными потребителями компактного компьютера будут школы и другие учебные заведения, в которых дети изучают программирование.

Пара лет ушла на создание различных вариантов устройства, пока к 2008 году не укрепилось понимание, что процессоры для мобильных устройств стали доступными и достаточно мощными для работы с медиаконтентом и именно их, а не микроконтроллеры, следует использовать для претворения идеи в жизнь. В 2009 году была создана благотворительная организация Raspberry Pi Foundation, в задачи которой входили разработка и продвижение компьютера Raspberry Pi. Еще два года потребовалось на создание аппаратной и программной части будущего устройства, заключение договоров и соблюдение прочих формальностей. И вот, в мае 2011 года британский программист и предприниматель Дэвид Брэбен (David John Braben) представил концепт компьютера Raspberry Pi, после чего были созданы альфа- и

бета-версии его плат. Но только в начале 2012 года первая партия Raspberry Pi отправилась на сборочный конвейер, а до заказчиков она добралась ближе к лету, потому что китайский подрядчик умудрился ошибиться при сборке, что вызвало дополнительные затраты времени на исправление ошибки.

Пробная партия Raspberry Pi объемом 10 тыс. экземпляров разошлась за несколько минут, причем поначалу действовало правило "одна штука в одни руки". Год спустя, во время старта продаж в США, история повторилась. А за полтора года, истекших после запуска, продано более полутора миллионов устройств, и это, похоже, не предел. Лицензией на производство плат обладают компании Premier Farnell, RS Components и Egoman. Причем последняя выпускает платы красного цвета, которые могут предлагаться только на китайских территориях. К первой годовщине проекта компанией RS Components была выпущена юбилейная партия плат синего цвета объемом 1000 штук. Указанные компании имеют право и продавать Raspberry Pi, а в США его распространением занимается компания Allied Electronics. Соответственно, все остальные магазины попросту закупают большие партии устройств у этой четверки и перепродают конечным потребителям. Обе модели плат от разных производителей (сборкой занимаются заводы Sony, Qisda и Egoman) имеют некоторые несущественные различия, но, по большому счету, они идентичны.

1.2. Технические характеристики и возможности

Raspberry Pi, как уже отмечалось ранее, представляет собой одноплатный компьютер размером с кредитную карту. На самом деле сама плата чуть крупнее: 85,6×56×21 мм — и не имеет скругленных краев, к тому же некоторые порты попросту торчат снаружи, не говоря уж про карту SD, которая более чем на половину выпирает за пределы платы. Весит устройство всего 54 грамма. Raspberry Pi выпускается в двух комплектациях: модель "A" и модель "B" (рис. 1.1). Сравнительные характеристики моделей "A" и "B" приведены в табл. 1.1.

Обе версии Raspberry Pi оснащены процессором Broadcom BCM2835 архитектуры ARM11 с тактовой частотой 700 МГц и модулем оперативной памяти на 256 (или 512) Мбайт, размещенным по технологии package-on-package[1] непосредственно на процессоре. Модель "A" обладает одним портом USB 2.0, тогда как модель "B" — двумя. У модели "B" наличествует и порт Ethernet. Помимо основного ядра, процессор BCM2835 включает в себя графическое ядро с поддержкой OpenGL ES 2.0, аппаратного ускорения и видео Full HD, а также ядро DSP (цифрового сигнального процессора).

Питание компьютера осуществляется через разъем micro-USB, при этом сила тока должна составлять минимум 0,5–0,7 А. При меньших значениях компьютер все еще

[1] Package on Package (PoP) — метод монтажа интегральных схем, когда один или более компонентов монтируются поверх друг друга (так называемый *вертикальный монтаж*). Эта технология значительно повышает плотность упаковки электронных компонентов на плате.

может включиться, но будет уходить в перезагрузку при запуске ресурсоемких задач. Следовательно, подключать плату лучше не через хаб, а напрямую к USB-порту компьютера или в розетку через специальный переходник.

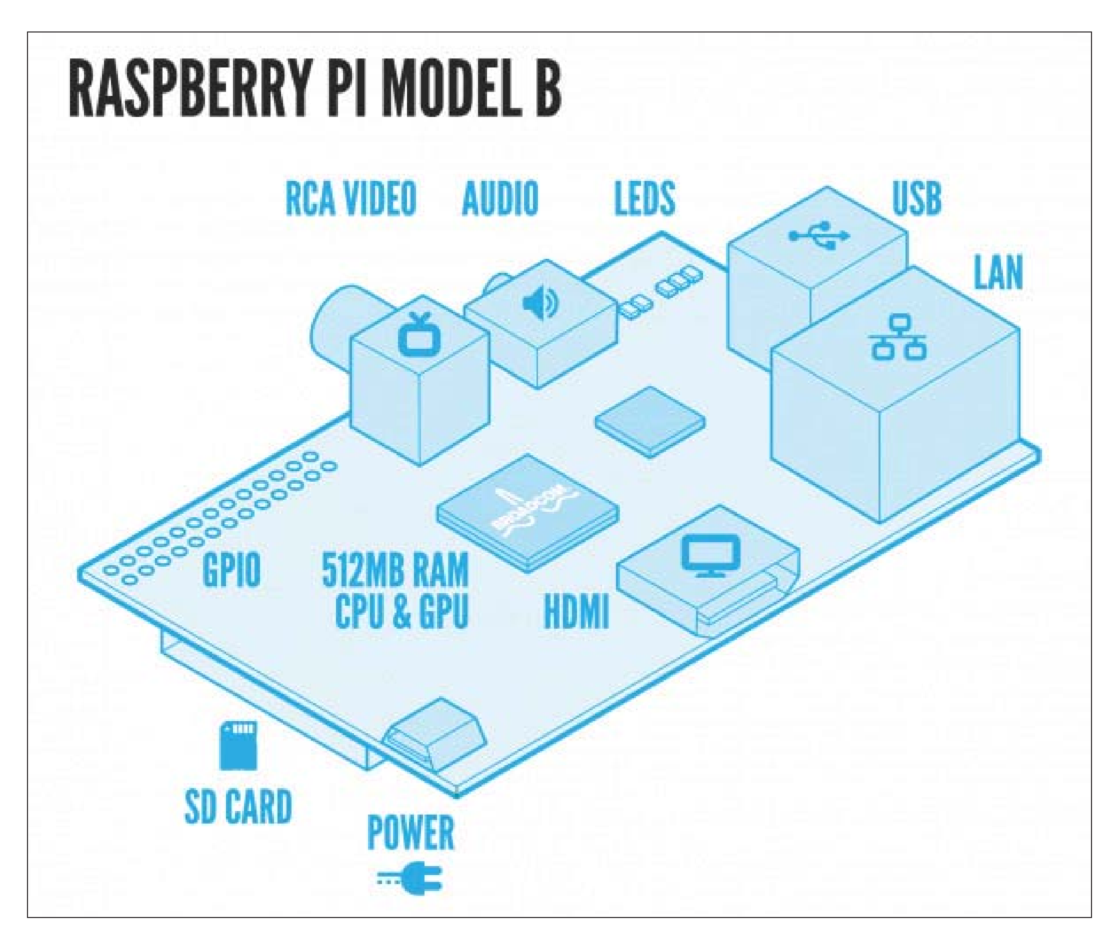


Рис. 1.1. Схема Raspberry Pi, модель "B"

Таблица 1.1. *Сравнительные характеристики моделей "A" и "B" Raspberry Pi*

Одноплатный мини-ПК Raspberry Pi		
Характеристики	**Model A**	**Model B**
Цена, доллары США	25	35
System-on-a-Chip (SoC)	Broadcom BCM2835 (CPU + GPU)	
CPU	700 МГц ARM11 (ядро ARM11 76JZF-S), возможен разгон до 1 ГГц	
GPU	Broadcom VideoCore IV	
Стандарты	OpenGL ES 1.1/2.0, OpenVG 1.1, Open EGL, OpenMAX	
Аппаратные кодеки	H.264(1080p30,high-profile); MPEG-2 и VC-1 (лицензия продается отдельно)	
Память (SDRAM, общая)	256 Мбайт	512 Мбайт; 256 Мбайт (до 15.10.2012)
Порты USB 2.0	1	2

Таблица 1.1 (окончание)

Характеристики	Model A	Model B
Видеовыход	1 × HDMI 1.3a (CEC), 1 × RCA (576i/480i, PAL-BGHID/M/N, NTSC, NTSC-J)	
Аудиовыход	Гнездо 3,5 мм, HDMI	
Кардридер	SD/MMC/SDIO	
Сеть	–	Ethernet-порт RJ45, 10/100 Мбит/с
Интерфейсы	20 × GPIO (SPI, I^{2}C, UART, TTL); MIPI CSI-2, MIPI DSI	
Энергопотребление	500 мА (2,5 Вт)	700 мА (3,5 Вт)
Питание	5 В через порт micro-USB или GPIO	
Размеры, мм	85,6×56×21	
Масса, г	54	

Никаких кнопок включения/выключения на корпусе платы не предусмотрено. Если необходимо запустить Raspberry Pi — подключаете USB-питание, для выключения — выдергиваете шнур. В будущих ревизиях, возможно, добавят питание по Ethernet, поскольку это один из самых частых запросов от пользователей.

Рядом с портом питания находится слот для SD-карт. Без карты памяти Raspberry Pi не включается, поскольку именно на ней записана операционная система, — это все равно что пробовать запустить компьютер без жесткого диска. Поскольку собственной ОС во внутренней памяти (как в телефонах) у Raspberry Pi не имеется, то из этого следует один положительный момент — устройство практически невозможно превратить в "кирпич". После любого неудачного эксперимента достаточно перезаписать дистрибутив на карту памяти, и Raspberry Pi снова заработает как новенький.

Выбор для карт памяти формата SD, а не Micro SD, имеет как свои плюсы, так и минусы. С одной стороны, плату можно было сделать еще компактней, отдав предпочтение Micro SD, тем более что эти карты более распространены, чем обычные SD. С другой стороны, решающим фактором стало удобство использования компьютера. Дело в том, что вставленная SD-карта немного выглядывает наружу, так что ее можно схватить за край и легко вытянуть. Если же использовать Micro SD, то доступ к слоту памяти будет заблокирован в случаях, когда на Raspberry Pi надет корпус.

Для подключения дисплея имеются сразу два интерфейса: RCA Video (композитный) и HDMI. Применяя переходники, можно выйти и на более традиционные VGA и DVI. HDMI поддерживает передачу как видео, так и звука, а если потребуется отдельный аудиоканал, то и он присутствует на плате в виде стандартного мини-джека 3,5 мм. Подключение микрофона также возможно, но для этого понадобится найти совместимое с Raspberry Pi USB-устройство.

Текущая модель Raspberry Pi не имеет модуля Wi-Fi, и для работы в Интернете понадобится задействовать порт Ethernet. Поскольку физически он скоммутирован через USB 2.0, то обеспечивает не гигабитную, а 100-мегабитную скорость.

На плате присутствуют всего два USB-порта — соответственно, после подключения клавиатуры и мыши свободных разъемов для флешек, USB-дисков, Wi-Fi/3G-донглов и других устройств не остается. Тут можно посоветовать либо подобрать клавиатуру с USB-хабом, либо докупить дополнительный хаб самостоятельно.

Чипы процессора и GPU не оснащены даже простейшими радиаторами, и после нескольких часов работы компьютера становится очевидным, почему остановились именно на этом решении, — плата нагревается при работе совсем незначительно, она скорее теплая, чем горячая.

Частота процессора составляет 700 МГц (ARM 11), и в зависимости от дистрибьютора процессор можно разогнать до 1000 МГц без потери гарантии (возможен выбор и более щадящих режимов). Чип памяти производства Samsung или Hynix напаян прямо поверх основного чипсета, так что увеличить RAM самостоятельно не получится. При покупке стоит обратить внимание на маркировку SoC (процессора System-on-a-Chip, системы на кристалле). Номер партии для "старых" версий модели "B" с 256 Мбайт RAM начинается с K4P2G, а у выпуска с 512 Мбайт памяти — с K4P4G.

Видеоускоритель Broadcom VideoCore IV позволяет даже при таком слабом процессоре декодировать видео 1080p h.264 с битрейтом вплоть до 40 Мбит/с. Для включения аппаратного ускорения MPEG-2 и VC-1 лицензии на применение этих технологий придется докупать отдельно.

Для визуальной индикации процессов плата оснащена пятью светодиодами. Три из них демонстрируют активность и режим работы Ethernet, а еще два сигнализируют о наличии питания и работе с SD-картой.

На рынке можно найти несколько корпусов — как официальных, так и сторонних производителей — для повышения защищенности компьютера и более удобной его транспортировки.

А теперь — самое интересное: набор низкоуровневых интерфейсов, которые позволяют подключать к Raspberry Pi платы расширения, внешние контроллеры, датчики и прочие аксессуары. Во-первых, на плате имеются 15-пиновые слоты CSI-2 для подключения камеры и DSI для установки дисплея. Во-вторых, присутствует колодка на 26 линий ввода/вывода общего назначения GPIO, из которых по факту для управления доступны только 17. На них же реализованы интерфейсы UART, консольный порт, шина SPI (Serial Peripheral Interface, последовательный периферийный интерфейс) и I^2C (Inter-Integrated Circuit, последовательная шина данных для связи интегральных схем). На новых ревизиях плат разведены, но не распаяны, еще четыре GPIO, дополнительно дающие I^2C и I^2S (Integrated Inter-chip Sound, последовательная шина данных, служащая для соединения цифровых аудиоустройств). Использование GPIO — это как раз самое интересное и творческое применение Raspberry Pi.

Впрочем, недостатков у Raspberry Pi тоже хватает. В нем, к примеру, нет собственных часов реального времени (Real Time Clock, RTC), поэтому единственный способ получения времени — это синхронизация с NTP-серверами. SoC Broadcom BCM2835 содержит в себе ядро цифрового сигнального процессора (DSP), но полного доступа к его API до сих пор нет. Выводы GPIO никак не защищены от короткого замыкания, поэтому ошибка в монтаже может сгубить весь мини-ПК. Кроме того, Raspberry Pi способен обрабатывать только цифровые сигналы. Видеовыходы не могут одновременно выводить картинку. Аудиовхода вообще нет...

Raspberry Pi может стать в ваших руках и медиацентром, и управляющим центром "умного дома", и сердцем робота. Тут уж все зависит от вашей фантазии и желания. В Сети есть немало примеров, готовых проектов, сообществ пользователей и целых магазинов, посвященных Raspberry Pi. Есть даже официальный очень-очень скромный интернет-магазин The Pi Store (**http://store.raspberrypi.com/projects**) с небольшим количеством ПО, игр, руководств и собственным журналом.

ГЛАВА 2



Установка ОС на Raspberry Pi

2.1. Дистрибутивы Raspberry Pi

Для того чтобы использовать Raspberry Pi, необходимо установить на него операционную систему. Доступно три официальных дистрибутива Linux:

- Pidora (основанный на Fedora);
- Archlinux (установка этого дистрибутива происходит практически вручную);
- Raspbian (основанный на Debian).

Кроме этих трех операционных систем, на Raspberry Pi портировано очень много других. Поскольку операционная система устанавливается на SD-карту, чтобы запустить другую систему, достаточно вставить в устройство карту с этой системой. В табл. 2.1 представлены некоторые дистрибутивы для Raspberry Pi.

Таблица 2.1. *Список операционных систем для Raspberry Pi*

№	Система	Описание, возможности	Сайт проекта
1	Raspbian	Официально рекомендуемый дистрибутив для использования на Raspberry Pi. Основан на пакетной базе Debian Wheezy и специально оптимизирован для Raspberry Pi (сборка для ARMv6 с расширениями "hard float")	**http://www.raspberrypi.org/downloads**
2	Raspbian Server Edition	Облегченная версия Raspbian для использования в качестве сервера	**http://www.sirlagz.net/?p=662**
3	Raspbian Minimal	Образ на основе урезанной Raspbian, который работает с sshd (демоном, предоставляющим защищенный доступ к ресурсам по протоколу OpenSSH) и минимальным набором установленных пакетов	**http://www.pi-point.co.uk/raspbian-minimal**
4	Pidora	Дистрибутив из пакетов Fedora для архитектуры ARMv6. Образ специально скомпилирован под оборудование мини-ПК Raspberry Pi и включает конфигурационные модули для упрощения его настройки под типичные задачи.	**http://pidora.ca/**

Таблица 2.1 (продолжение)

№	Система	Описание, возможности	Сайт проекта
		Дистрибутив по умолчанию включает в себя языки программирования C, Python и Perl, а также сопутствующие программные инструменты. Кроме того, присутствуют библиотеки с поддержки внешних аппаратных модулей, подключаемых через интерфейсы GPIO, I^2C и SPI, а также софт для робототехники	
5	Arch Linux ARM	Дистрибутив основан на Arch Linux, предназначен для опытных пользователей и позволяет получить полный контроль над ПК	**http://www.archlinuxarm.org/**
6	Raspbmc	Вариант системы, оптимизированный для запуска XBox Media Centre. Создан на основе Raspbian	**http://www.raspbmc.com/**
7	XBian	Небольшой, быстрый и легковесный дистрибутив Media Center для Raspberry Pi, основанный на минимальном образе Raspbian с XBMC	**http://www.xbian.org**
8	OpenELEC	Еще вариант оптимизированной операционной системы, предназначенный специально для запуска XBox Media Centre. В отличие от Raspbmc, OpenElec не содержит ничего, кроме необходимого минимума для выполнения основных ее функций. Поэтому достаточно трудно, например, установить другие пакеты и, как следствие, набор различных дополнений очень скуден. Но все это сделано для достижения высокой производительности	**http://www.openelec.thestateofme.com/**
9	DarkELEC	Этот ответвление от OpenELEC, устраняющее многие недостатки родителя и сосредоточенное на 100%-ной поддержке Raspberry Pi	**http://www.darkimmortal.com/category/raspberry-pi/**
10	Plan 9	UNIX-подобная операционная система от Bell Labs с открытым исходным кодом, имеющая очень простой пользовательский интерфейс и поддерживающая имена файлов в кодировке UTF-8	**http://www.plan9.bell-labs.com/plan9/**
11	Risc OS	Операционная система, разработанная компанией Acorn Computers для серии своих настольных компьютеров, использующих центральный процессор архитектуры ARM	**http://www.riscosopen.org/content/**
12	Raspberry Pi Thin Client	Тонкий клиент — поддержка Microsoft RDC, Citrix ICA, VMWare View, OpenNX и SPI CE на платформе Raspberry Pi	**http://www.rpitc.blogspot.se/**
13	Pi Point	Включает Raspberry Pi как беспроводную точку доступа	**http://www.rpitc.blogspot.se/**
14	Coder	Google выпустил дистрибутив Coder, который позволяет запустить на Raspberry Pi простой веб-сервер, предоставляющий доступ к специализированной среде разработки, нацеленной на обучение созданию веб-приложений, написанных на HTML, CSS и JavaScript	**http://googlecreativelab.github.io/coder/**

Таблица 2.1 *(окончание)*

№	Система	Описание, возможности	Сайт проекта
15	Android	Компания Broadcom работала над официальной сборкой Android 4. Однако на данный момент можно скачать сборку Android 2.3, которая, к сожалению, слишком медленная, чтобы быть полезной	**http://androidpi.wikia.com/wiki/Android_Pi_Wiki**
16	piCore	Минималистичный дистрибутив GNU/Linux, цель которого — предоставление базовой системы с использованием BusyBox, FLTK и другого легковесного программного обеспечения. Размер дистрибутива около 10 мегабайт, устанавливается в текстовом режиме	**http://tinycorelinux.net/5.x/armv6/release/**

Количество операционных систем для Raspberry Pi постоянно растет. В настоящее время их более 30, и сообщество Linux постоянно создает что-то новое.

2.2. Установка ОС с помощью NOOBS

Самый простой способ установить дистрибутив на Raspberry Pi — инструмент NOOBS от создателей Raspberry Pi, который позволяет при первой загрузке выбрать для установки одну операционную систему из следующих:

- ArchLinux;
- OpenElec;
- Pidora;
- Raspbmc;
- Raspbian;
- Risc Os.

Для установки дистрибутива для Raspberry Pi с помощью NOOBS потребуется SD-карта емкостью 4 Гбайт или более. SD-карту необходимо отформатировать, в чем нам поможет программа SD Formatter 4.0, архив которой доступен для закачки по ссылке: **https://www.sdcard.org/downloads/formatter_4/eula_windows/** (рис. 2.1).

Итак, скачиваем программу, устанавливаем ее и запускаем. Выбираем нашу SD-карту, нажав кнопку **Options**, задаем параметр **FORMAT SIZE ADJUSTMENT** равным **ON** и форматируем (рис. 2.2).

Далее скачиваем программное обеспечение NOOBS (**http://downloads.raspberrypi.org/noobs**) и распаковываем ZIP-файл на нашу SD-карту.

По завершении записи вынимаем SD-карту из кардридера компьютера и вставляем ее в Raspberry Pi. Подключаем монитор (по HDMI или VGA), клавиатуру, а также опционально мышь и кабель Ethernet. Затем подаем питание через порт micro-USB. При первой загрузке видим меню, предлагающее установить одну из нескольких операционных систем в свободное пространство на карте памяти (рис. 2.3).



Рис. 2.1. Страница загрузки программы SD Formatter 4.0

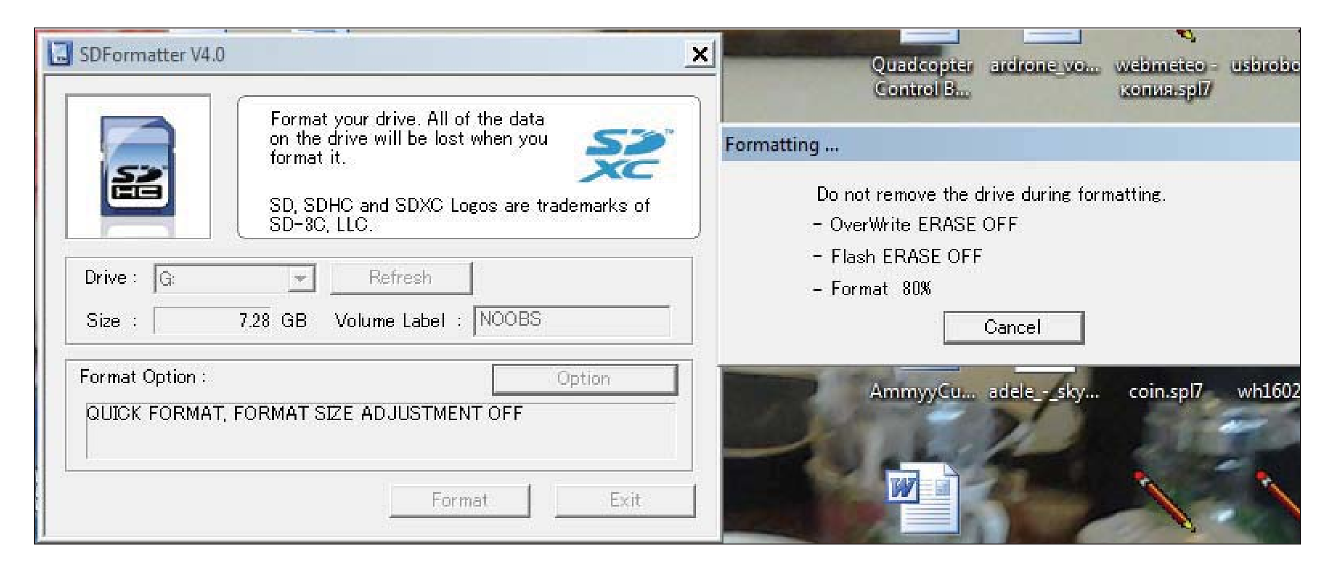


Рис. 2.2. Окно программы SD Formatter 4.0

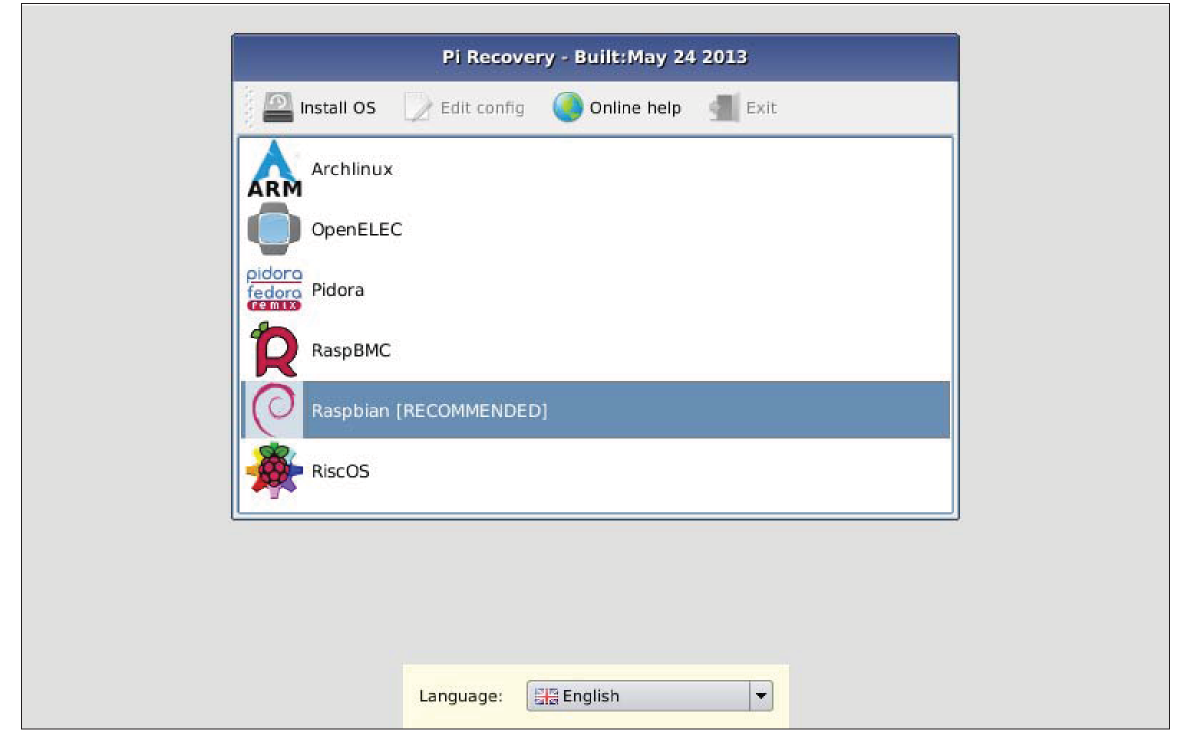


Рис. 2.3. Меню NOOBS при первой загрузке

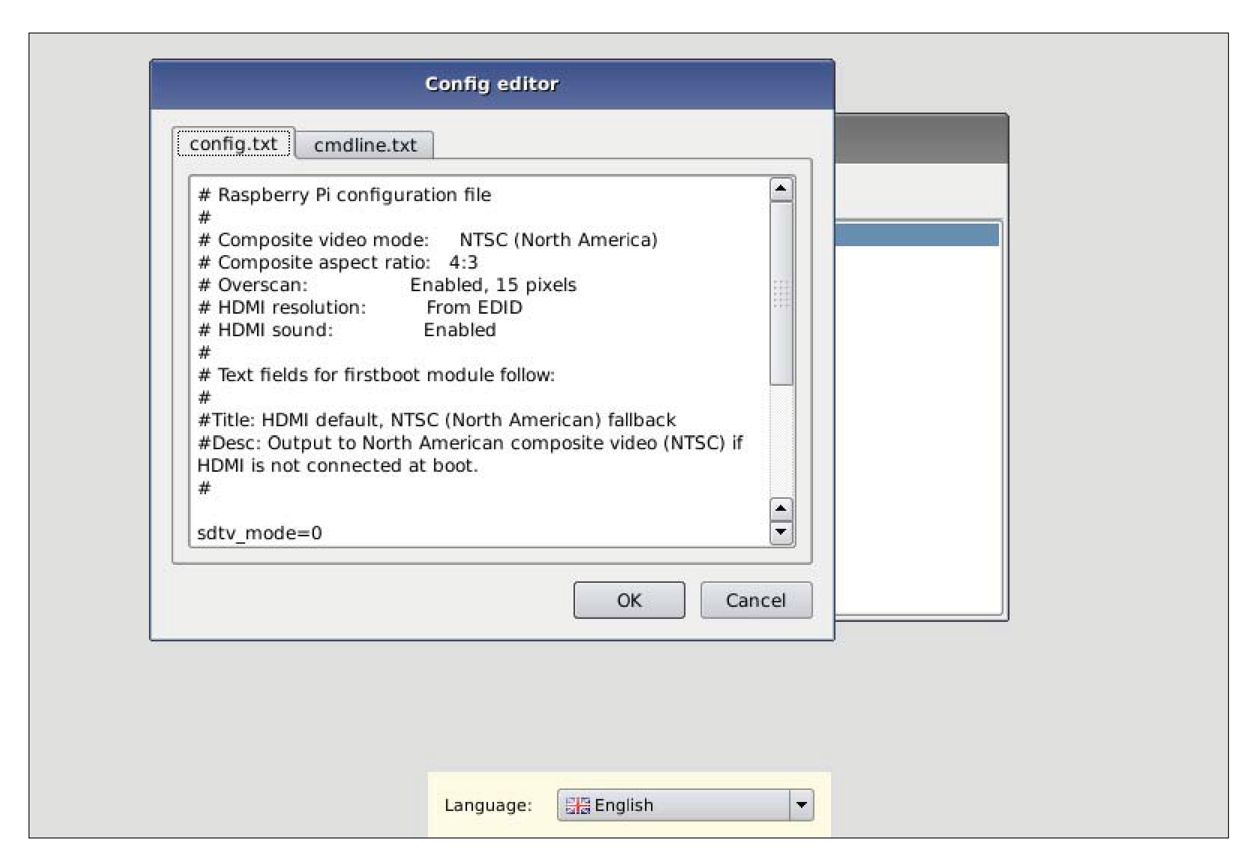


Рис. 2.4. Редактирование файла config.txt

Raspberry Pi • Index page - Arora

File Edit View History Bookmarks Window Tools Help

http://www.raspberrypi.org/phpBB3/

Raspberry Pi • In...

Forum	Topics	Posts	Last post	When
Basic setup and usage	3623	24025	by Joe Schmoe	Tue May 28, 2013 12:48 pm
New Linux user help	1194	7302	by richddean	Tue May 28, 2013 12:45 pm
Troubleshooting	4926	33719	by elagil	Tue May 28, 2013 12:58 pm
Power users	648	5912	by lewzer	Tue May 28, 2013 7:22 am
What software?	472	2805	by joan	Tue May 28, 2013 12:27 pm

Education

Forum	Topics	Posts	Last post	When

Рис. 2.5. Встроенный браузер Arora

После установки операционной системы Raspberry Pi будет загружаться в обычном режиме. Тем не менее NOOBS остается на карте, поэтому, удерживая клавишу <Shift> во время загрузки устройства, можно вернуться к меню выбора. Это позволяет переключиться на другую операционную систему или заново переписать установку текущей, а также предоставляет удобный инструмент для редактирования установленной операционной системы — файл конфигурации config.txt (рис. 2.4) и даже веб-браузер (рис. 2.5), так что в случае проблем с установкой можно будет найти информацию в Сети на тематических форумах.

2.3. Установка дистрибутива Raspbian с помощью загрузочной карты

Рассмотрим процесс установки дистрибутива Raspbian с помощью файла образа, записанного на SD-карту. Сначала скачиваем сам образ. Дистрибутив Raspbian wheezy, необходимый для работы Raspberry Pi, можно найти на официальном сайте проекта **www.raspberrypi.org** в разделе **Downloads** (рис. 2.6).

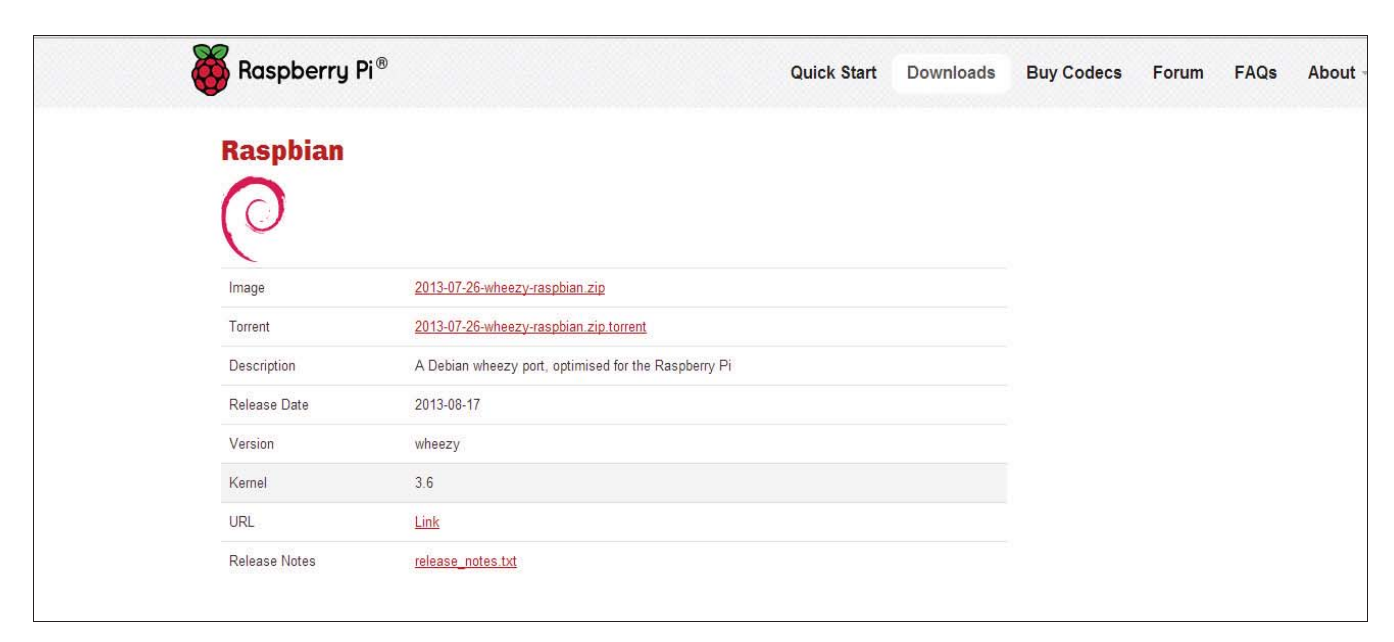


Рис. 2.6. Страница проекта **Raspberrypi.org** для скачивания образа Raspbian

Для записи образа Raspbian на SD-карту воспользуемся программой Win32 Disk Imager, скачать которую можно по ссылке: **http://sourceforge.net/projects/win32diskimager/files/latest/download?source=navbar**. Выбираем нужный образ и записываем на карту (рис. 2.7).

Затем вставляем карту в Raspberry Pi, включаем его и дожидаемся появления меню конфигурации **raspi-config** (рис. 2.8).

Рассмотрим некоторые пункты меню:

- **Expand Filesystem** — расширение раздела на все пространство флеш-накопителя, операция будет выполнена после перезагрузки;
- **Change User Password** — изменение пароля пользователя Pi;

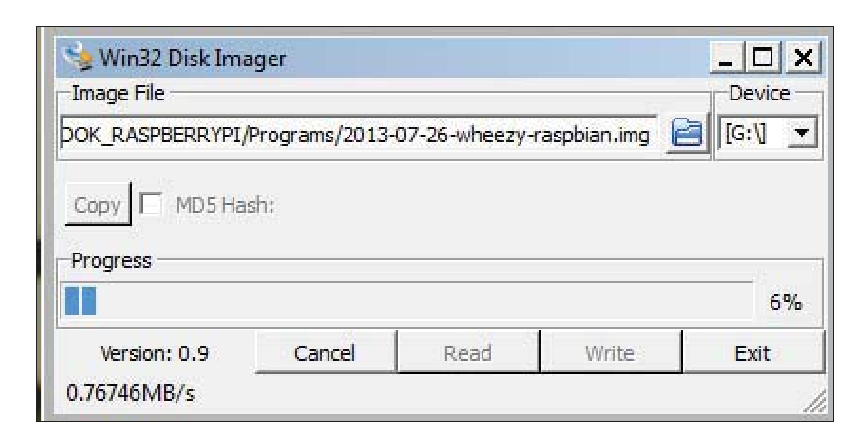


Рис. 2.7. Запись образа с помощью программы **Win32 Disk Imager**

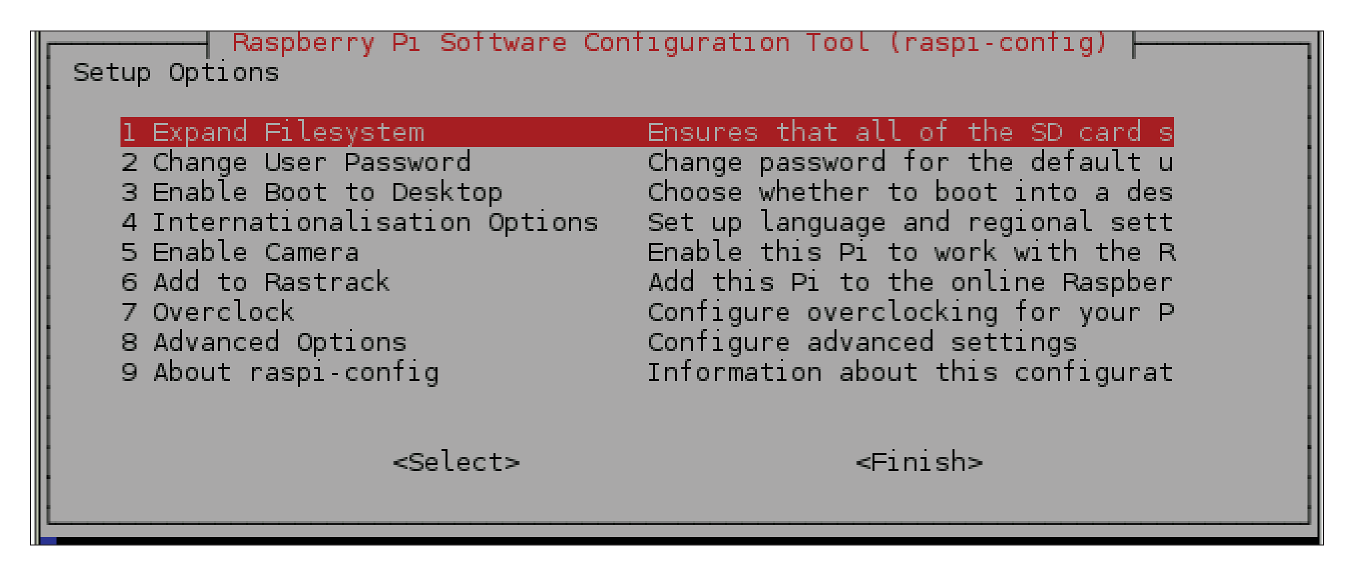


Рис. 2.8. Меню конфигурации Raspbian

- **Enable Boot to Desktop** — запуск графического режима при загрузке;
- **Internationalisation Options** — выбор языка и региональных настроек:
  - **Change locale** — изменение языка. Устанавливаем два значения: **en_GB.UTF-8** и **ru_RU.UTF-8**;
  - **Change Timezone** — настройка часового пояса;
  - **Change Keyboard Layout** — изменение раскладки клавиатуры;
- **Enable Camera** — поддержка модуля камеры (рис. 2.9);
- **Overclock** — увеличение частоты процессора (разгон). Можно разогнать до 1 ГГц;
- **Advanced Options** — дополнительные параметры:
  - **Overscan** — настройка режима overscan (вылета развертки). Если по краю изображения имеется широкая черная полоса, то необходимо выключить (**Disable**) этот режим;
  - **Hostname** — имя компьютера в сети, по умолчанию raspberrypi;
  - **Memory split** — количество памяти, выделяемое под видео. По умолчанию — 64 Мбайт;

- **SSH** — включение ssh-сервера;
- **Update** — обновление программы raspi-config;

❒ **About raspi-config** — информация о программе.

Завершив настройку, выбираем **Finish**. Система запросит разрешение на перезагрузку. Соглашаемся. Если вы позже захотите поменять какие-то настройки, необходимо будет в консоли набрать команду:

```
sudo raspi-config
```

После продолжительной перезагрузки система выйдет на запрос логина и пароля. Вводим логин `pi` и пароль, который был указан при настройке. Для запуска графического режима набираем в консоли команду:

```
startx
```

и попадаем на рабочий стол Raspbian (см. рис. 2.9).

На этом установка и первичная настройка системы завершены!

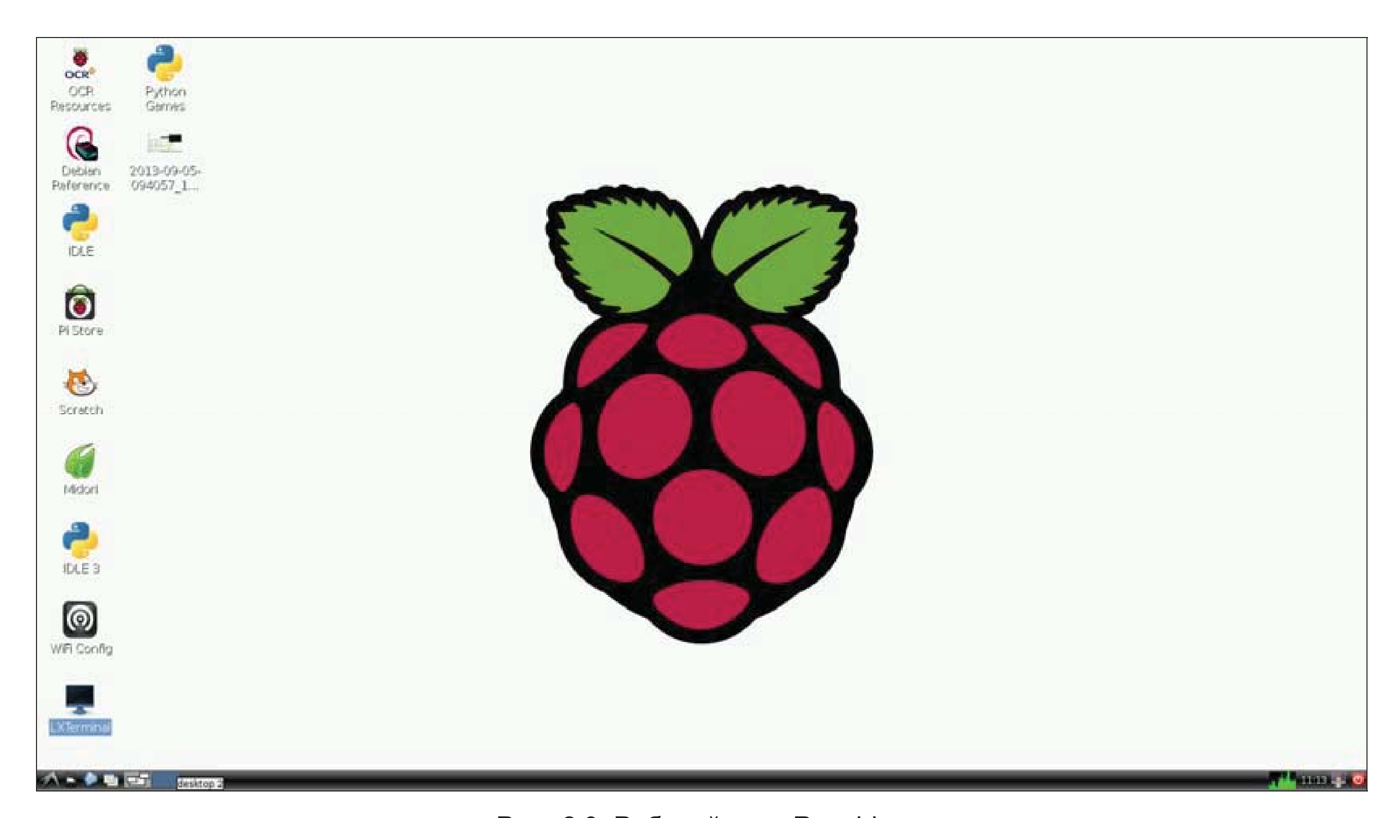


Рис. 2.9. Рабочий стол Raspbian



ГЛАВА 3

Дополнительное оснащение мини-ПК Raspberry Pi

3.1. Корпус

Стремясь сделать устройство как можно дешевле, разработчики выкинули все "лишнее". В этот разряд попал и корпус. Плата с торчащими во все стороны разъемами выглядит очень беззащитно, и ее хочется во что-нибудь спрятать. Корпусов под Raspberry Pi существует куча — разных, на любой вкус. Можно заказать и стандартный пластиковый корпус (рис. 3.1). Цены — в районе 10 долларов.

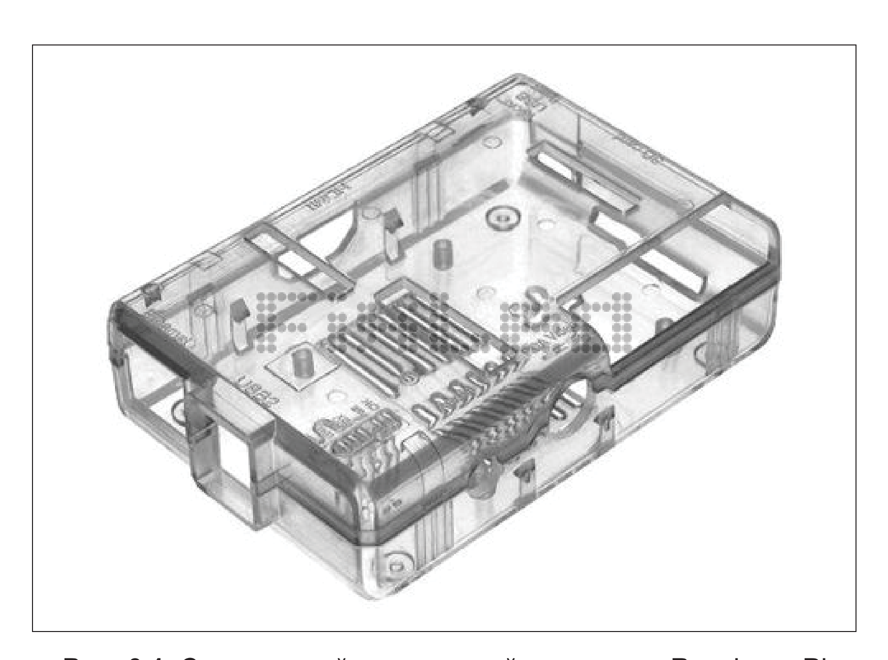


Рис. 3.1. Стандартный пластиковый корпус для Raspberry Pi

Корпус можно напечатать на 3D-принтере (соответствующие модели легко найти поиском в Интернете), можно выпилить его из оргстекла и даже собрать из конструктора Lego. Интернет-магазины также предлагают множество моделей, и некоторые из них весьма стильные (рис. 3.2 и 3.3). Так что найти корпус по своему вкусу не проблема. Но корпус желателен. С ним Raspberry Pi выглядит как полноценный компьютер, и главное — корпус выполняет защитные функции.



Рис. 3.2. Сборный цветной корпус для Raspberry Pi



Рис. 3.3. Еще один корпус для Raspberry Pi

3.2. Источник питания

Для Raspberry Pi требуется блок питания с током не менее 700 мА, и чем ток больше — тем лучше. Для своего первого Raspberry Pi, на который я установил дистрибутив Raspbmc, было использовано подзарядное устройство для телефона Samsung (рис. 3.4). И на одном из моих Raspberry Pi оно бесперебойно работает до сих пор.



Рис. 3.4. Зарядное устройство Samsung для Raspberry Pi

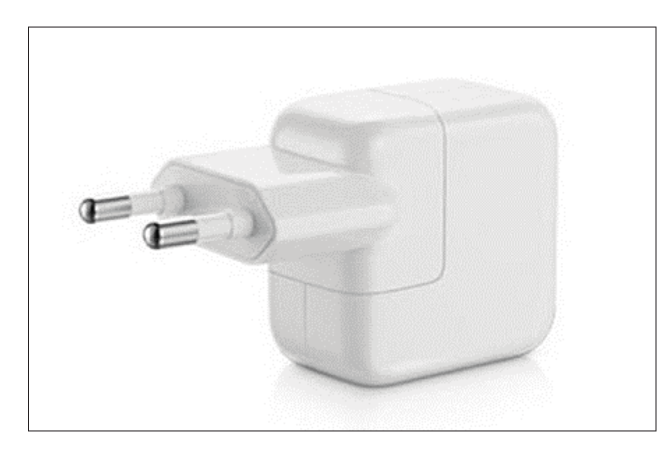


Рис. 3.5. 10-ваттный USB Power Adapter от iPhone, выдающий ток 2,1 А напряжением 5 В

К тому, первому Raspberry Pi, из внешних устройств были подключены только клавиатура и мышь. Если же использовать дополнительные устройства, подключаемые по USB (например, адаптер Wi-Fi или USB-камеру), потребуется более мощный источник питания — например, 10-ваттный USB Power Adapter от iPhone, выдающий ток 2,1 А напряжением 5 В (рис. 3.5).

Если не хватает USB-входов, и для расширения их количества вы используете какой-либо USB-концентратор (хаб) — например, 7-портовый Defender Hi-speed USB Hub (рис. 3.6), то при подключении через него внешнего USB-диска или нескольких устройств необходимо дополнительно запитывать этот USB-хаб через внешний источник питания.



Рис. 3.6. USB-хаб с внешним источником питания

3.3. Клавиатура и мышь

Для работы необходимы клавиатура и мышь. Устройства с разъемом PS/2 не подойдут. Задействовать для них два USB-входа (на клавиатуру и на мышь) нецелесообразно, поэтому лучше приобрести беспроводной набор из мыши и клавиатуры. Перед покупкой необходимо убедиться, что клавиатура входит в список совместимого оборудования (**http://elinux.org/RPi_VerifiedPeripherals**). Я в качестве устройства ввода использовал мини-клавиатуру со встроенным тачпадом (рис. 3.7).



Рис. 3.7. Беспроводная мини-клавиатура со встроенным тачпадом

3.4. Монитор

Есть несколько вариантов подключения Raspberry Pi к монитору или телевизору. Прежде всего, это порт HDMI, который обеспечивает вывод цифровых видео- и аудиосигналов с поддержкой 14 различных разрешений видео.

Видеосигнал может быть передан также на вход DVI (стандартный для многих мониторов), при этом для подключения к монитору понадобится переходник HDMI-DVI (рис. 3.8).



Рис. 3.8. Переходник HDMI-DVI

Еще один вариант вывода видеосигнала — стандартный разъем RCA ("тюльпан"), на который выводится композитный видеосигнал в стандарте NTSC или PAL. По сравнению со стандартом HDMI этот выход обеспечивает гораздо более низкое разрешение. В продаже имеются маленькие мониторы, подключаемые по входу RCA. Вот и я приобрел себе монитор 4,3" с разрешением 800×480 пикселов (рис. 3.9) по цене всего 25 долларов (в Китае на сайте **aliexpress.com**).



Рис. 3.9. Монитор 4,3" со входом RCA для Raspberry Pi



Рис. 3.10. Преобразователь видео HDMI-VGA

Для подключения к Raspberry Pi VGA-мониторов необходим преобразователь, HDMI-VGA (рис. 3.10), купить который также можно в Китае по цене до 10 долларов.

Если при подключении монитора или телевизора по HDMI Raspberry Pi автоматически определяет разрешение экрана, то при подключении VGA необходимо вручную внести в файл config.txt, находящийся на карте памяти, следующие изменения:

```
# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1
# uncomment to force a specific HDMI mode (this will force VGA)
hdmi_group=2
hdmi_mode=16
# uncomment to force a HDMI mode rather than DVI. This can make audio work in
# DMT (computer monitor) modes
hdmi_drive=2
```

Параметр `hdmi_mode=16` определяет разрешение экрана и частоту развертки, и его точное значение можно подобрать экспериментально. Перечень всех опций файла config.txt можно посмотреть в Интернете по адресу **http://elinux.org/RPiconfig**, а список разрешений для параметра `hdmi_mode` представлен в листинге 3.1.

Листинг 3.1. Список разрешений для параметра `hdmi_mode`

```
hdmi_mode=1  640x350 85Hz
hdmi_mode=2  640x400 85Hz
hdmi_mode=3  720x400 85Hz
hdmi_mode=4  640x480 60Hz
hdmi_mode=5  640x480 72Hz
hdmi_mode=6  640x480 75Hz
hdmi_mode=7  640x480 85Hz
hdmi_mode=8  800x600 56Hz
hdmi_mode=9  800x600 60Hz
hdmi_mode=10 800x600 72Hz
hdmi_mode=11 800x600 75Hz
hdmi_mode=12 800x600 85Hz
hdmi_mode=13 800x600 120Hz
```

```
hdmi_mode=14 848x480 60Hz
hdmi_mode=15 1024x768 43Hz DO NOT USE
hdmi_mode=16 1024x768 60Hz
hdmi_mode=17 1024x768 70Hz
hdmi_mode=18 1024x768 75Hz
hdmi_mode=19 1024x768 85Hz
hdmi_mode=20 1024x768 120Hz
hdmi_mode=21 1152x864 75Hz
hdmi_mode=22 1280x768    reduced blanking
hdmi_mode=23 1280x768 60Hz
hdmi_mode=24 1280x768 75Hz
hdmi_mode=25 1280x768 85Hz
hdmi_mode=26 1280x768 120Hz reduced blanking
hdmi_mode=27 1280x800    reduced blanking
hdmi_mode=28 1280x800 60Hz
hdmi_mode=29 1280x800 75Hz
hdmi_mode=30 1280x800 85Hz
hdmi_mode=31 1280x800 120Hz reduced blanking
hdmi_mode=32 1280x960 60Hz
hdmi_mode=33 1280x960 85Hz
hdmi_mode=34 1280x960 120Hz reduced blanking
hdmi_mode=35 1280x1024 60Hz
hdmi_mode=36 1280x1024 75Hz
hdmi_mode=37 1280x1024 85Hz
hdmi_mode=38 1280x1024 120Hz reduced blanking
hdmi_mode=39 1360x768 60Hz
hdmi_mode=40 1360x768 120Hz reduced blanking
hdmi_mode=41 1400x1050   reduced blanking
hdmi_mode=42 1400x1050 60Hz
hdmi_mode=43 1400x1050 75Hz
hdmi_mode=44 1400x1050 85Hz
hdmi_mode=45 1400x1050 120Hz reduced blanking
hdmi_mode=46 1440x900    reduced blanking
hdmi_mode=47 1440x900 60Hz
hdmi_mode=48 1440x900 75Hz
hdmi_mode=49 1440x900 85Hz
hdmi_mode=50 1440x900 120Hz reduced blanking
hdmi_mode=51 1600x1200 60Hz
hdmi_mode=52 1600x1200 65Hz
hdmi_mode=53 1600x1200 70Hz
hdmi_mode=54 1600x1200 75Hz
hdmi_mode=55 1600x1200 85Hz
hdmi_mode=56 1600x1200 120Hz reduced blanking
hdmi_mode=57 1680x1050   reduced blanking
hdmi_mode=58 1680x1050 60Hz
hdmi_mode=59 1680x1050 75Hz
hdmi_mode=60 1680x1050 85Hz
```

```
hdmi_mode=61 1680x1050 120Hz reduced blanking
hdmi_mode=62 1792x1344 60Hz
hdmi_mode=63 1792x1344 75Hz
hdmi_mode=64 1792x1344 120Hz reduced blanking
hdmi_mode=65 1856x1392 60Hz
hdmi_mode=66 1856x1392 75Hz
hdmi_mode=67 1856x1392 120Hz reduced blanking
hdmi_mode=68 1920x1200   reduced blanking
hdmi_mode=69 1920x1200 60Hz
hdmi_mode=70 1920x1200 75Hz
hdmi_mode=71 1920x1200 85Hz
hdmi_mode=72 1920x1200 120Hz reduced blanking
hdmi_mode=73 1920x1440 60Hz
hdmi_mode=74 1920x1440 75Hz
hdmi_mode=75 1920x1440 120Hz reduced blanking
hdmi_mode=76 2560x1600   reduced blanking
hdmi_mode=77 2560x1600 60Hz
hdmi_mode=78 2560x1600 75Hz
hdmi_mode=79 2560x1600 85Hz
hdmi_mode=80 2560x1600 120Hz reduced blanking
hdmi_mode=81 1366x768 60Hz
hdmi_mode=82 1080p  60Hz
hdmi_mode=83 1600x900    reduced blanking
hdmi_mode=84 2048x1152   reduced blanking
hdmi_mode=85 720p   60Hz
hdmi_mode=86 1366x768    reduced blanking
```

3.5. Увеличение тактовой частоты (разгон)

Мини-компьютер Raspberry Pi с момента выпуска поддерживает оверклокинг и овервольтинг процессора с повышением, соответственно, его тактовой частоты и напряжения, что можно осуществить корректировкой опций файла config.txt. Это, однако, приводит к излишнему нагреву модуля, поэтому изначально было сказано, что разгон процессора нарушает условия гарантии. Тем не менее, разработчики провели тщательные испытания и создали схему оверклокинга с повышением напряжения, при которой они готовы сохранить гарантию. Это и есть "официальный турборежим" под контролем драйвера `cpufreq`.

Официальный турборежим предусматривает разгон процессора в моменты максимальной нагрузки и снижение частоты при повышении его температуры до 85 °C. В этом случае компьютеру Raspberry Pi ничего не угрожает на протяжении всего срока его эксплуатации.

В меню конфигурации дистрибутива Raspbian **raspi-config** (рис. 3.11) предлагается пять шаблонов оверклокинга, максимальный из которых разгоняет ARM-процессор до 1 ГГц. В этом режиме процессор показывает производительность целочисленных вычислений на 52 % выше, чем в стандартном режиме 700 МГц, операции

с плавающей точкой в тесте nbench выполняются на 64 % быстрее, а тесты памяти показывают прибавку 55 %.

В случае, если из-за слишком больших установок оверклокинга компьютер отказывается загружаться, надо при его загрузке удерживать клавишу <Shift> — тогда настройки сбросятся до дефолтных, и можно будет выбрать уровень поменьше.

```
Chose overclock preset

None   700MHz ARM, 250MHz core, 400MHz SDRAM, 0 overvolt
Modest 800MHz ARM, 250MHz core, 400MHz SDRAM, 0 overvolt
Medium 900MHz ARM, 250MHz core, 450MHz SDRAM, 2 overvolt
High   950MHz ARM, 250MHz core, 450MHz SDRAM, 6 overvolt
Turbo  1000MHz ARM, 500MHz core, 600MHz SDRAM, 6 overvolt

            <Ok>                    <Отмена>
```

Рис. 3.11. Шаблоны оверклокинга в меню конфигурации **raspi-config**



Рис. 3.12. Установка пассивных радиаторов

Особо горячие места на плате Raspberry Pi — это сам процессор, Ethernet/USB-контроллер и стабилизатор напряжения питания. Для уменьшения температуры можно установить на эти элементы пассивные радиаторы (рис. 3.12), "посадив" их на термоклей. Такое охлаждение позволит поддерживать температуру процессора при работе в десктопном режиме и с использованием программ в районе 50 °С. Посмотреть температуру платы (в миллиградусах Цельсия) можно в терминале командой:

```
cat /sys/class/thermal/thermal_zone0/temp
```

3.6. Подключение USB-накопителя

В качестве носителя информации в Raspberry Pi можно использовать USB-флешку или внешний жесткий диск. Для жесткого диска следует организовать отдельное питание, поскольку USB-порты платы на такие нагрузки не рассчитаны, и, в лучшем случае, жесткий диск просто не определится. Итак, подключаем USB-флешку и выполняем команду:

```
sudo fdisk -l
```

Команда покажет все носители, которые подключены к нашему Raspberry Pi (рис. 3.13). Искомый путь к устройству: **/dev/sda1**.

```
pi@raspberrypi1 ~ $ sudo fdisk -l

Disk /dev/mmcblk0: 3980 MB, 3980394496 bytes
4 heads, 16 sectors/track, 121472 cylinders, total 7774208 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00047c7a

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880     7774207     3825664   83  Linux

Disk /dev/sda: 16.0 GB, 16013852672 bytes
78 heads, 14 sectors/track, 28641 cylinders, total 31277056 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xc3072e18

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1            8064    31277055    15634496    c  W95 FAT32 (LBA)
pi@raspberrypi1 ~ $
```

Рис. 3.13. Список подключенных носителей

Для форматирования носителя запускаем программу fdisk (рис. 3.14):

```
sudo fdisk /dev/sda1
```

Вначале командой `d` удаляем существующие разделы (нужные разделы выбираем цифрами), затем создаем новый раздел с помощью команды `n` (все значения принимаем по умолчанию) и сохраняем проделанную работу с помощью команды `w`.

Создаем на носителе файловую систему ext2:

```
sudo mkfs -t ext2 /dev/sda1
```

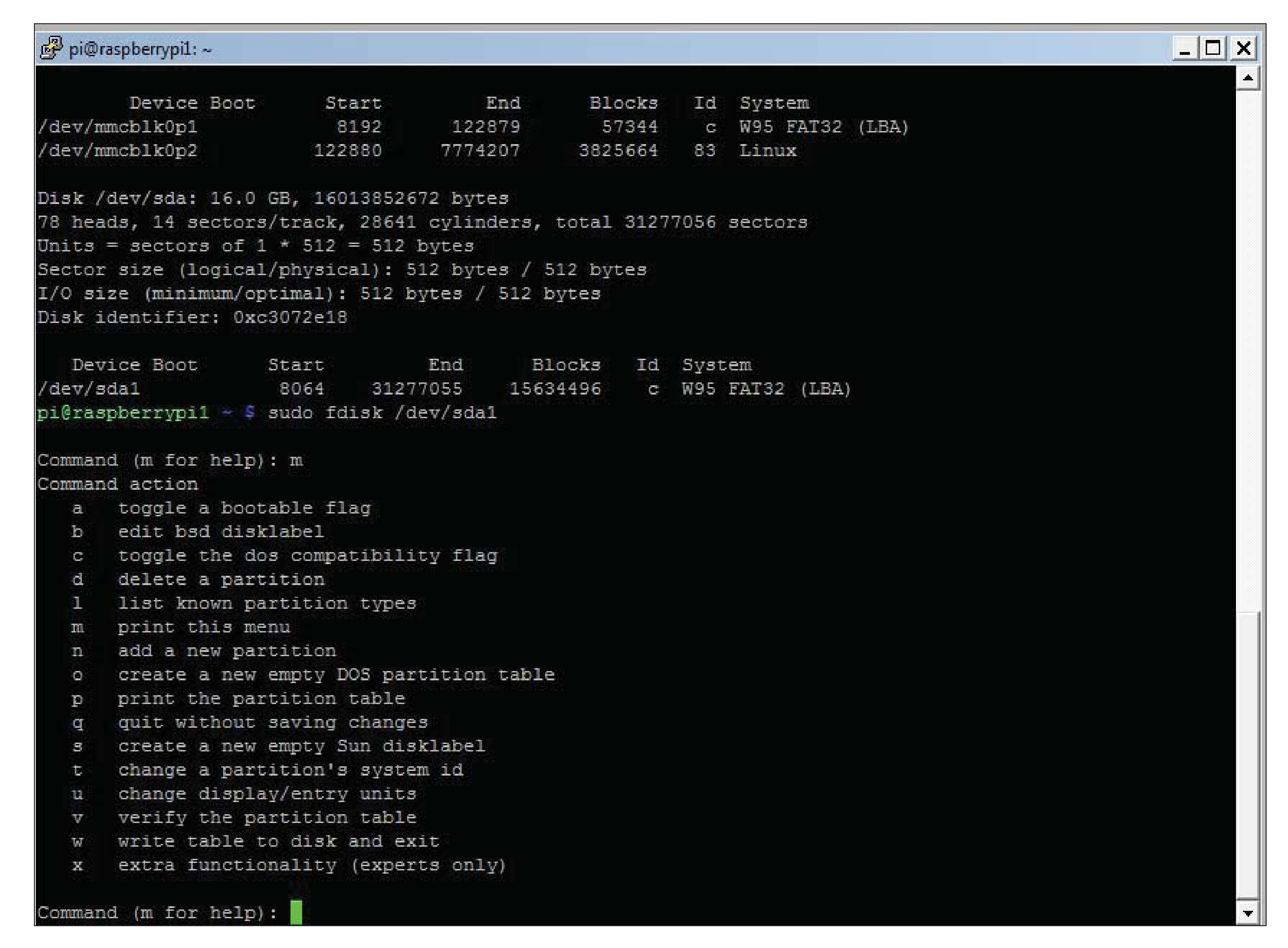


Рис. 3.14. Запуск программы fdisk

Монтируем:

```
sudo mount -t ext2 /dev/sda1
```

Чтобы обеспечить автоматическое монтирование носителя при каждой загрузке Raspbian, создаем папку:

```
sudo mkdir /mnt/flash
sudo chmod -R 777 /mnt/flash
```

Открываем файл настроек:

```
sudo nano /etc/fstab
```

и добавляем в него строку:

```
/dev/sda1 /mnt/flash ext2 defaults,auto,umask=000,users,rw 0 0
```

Сохраняем и перезагружаем устройство. Теперь при загрузке носитель должен автоматически примонтироваться, что можно проверить командой:

```
df
```

Она выведет список примонтированных устройств с указанием точек их монтирования (рис. 3.15).

```
pi@raspberrypi1 ~ $ df
Файловая система 1K-блоков Использовано Доступно Использовано% Смонтировано в
rootfs             3763920      1737852  1854796           49% /
/dev/root          3763920      1737852  1854796           49% /
devtmpfs             53720            0    53720            0% /dev
tmpfs                12396          392    12004            4% /run
tmpfs                 5120            0     5120            0% /run/lock
tmpfs                24780            0    24780            0% /run/shm
/dev/mmcblk0p1       57288        18888    38400           33% /boot
/dev/sda1         15630400      1516800 14113600           10% /mnt/flash
pi@raspberrypi1 ~ $
```

Рис. 3.15. Список примонтированных устройств

3.7. Подключение жесткого диска

Если вам нужен значительный объем дискового пространства — например, для хранения фильмов, объема флеш-накопителя USB не хватит. Необходимо подключать внешний жесткий диск. Для этого мы воспользуемся, как уже отмечалось ранее, USB-хабом с внешним источником питания Defender Hi-speed USB Hub (см. рис. 3.6) и подключим через него переносной жесткий диск WD 320 Гбайт (рис. 3.16) с файловой системой NTFS.



Рис. 3.16. Внешний жесткий диск WD 320 Гбайт

В дистрибутиве Raspbian он монтируется как read-only (только для чтения), что нас не устраивает. Для поддержки записи на NTFS из Linux установим нужные драйверы:

```
sudo apt-get install ntfs-3g
```

Далее узнаем имя раздела для монтирования:

```
sudo fdisk -l
```

Из результата вывода (рис. 3.17) заключаем, что внешний HDD в данном случае имеет имя **sda1**.

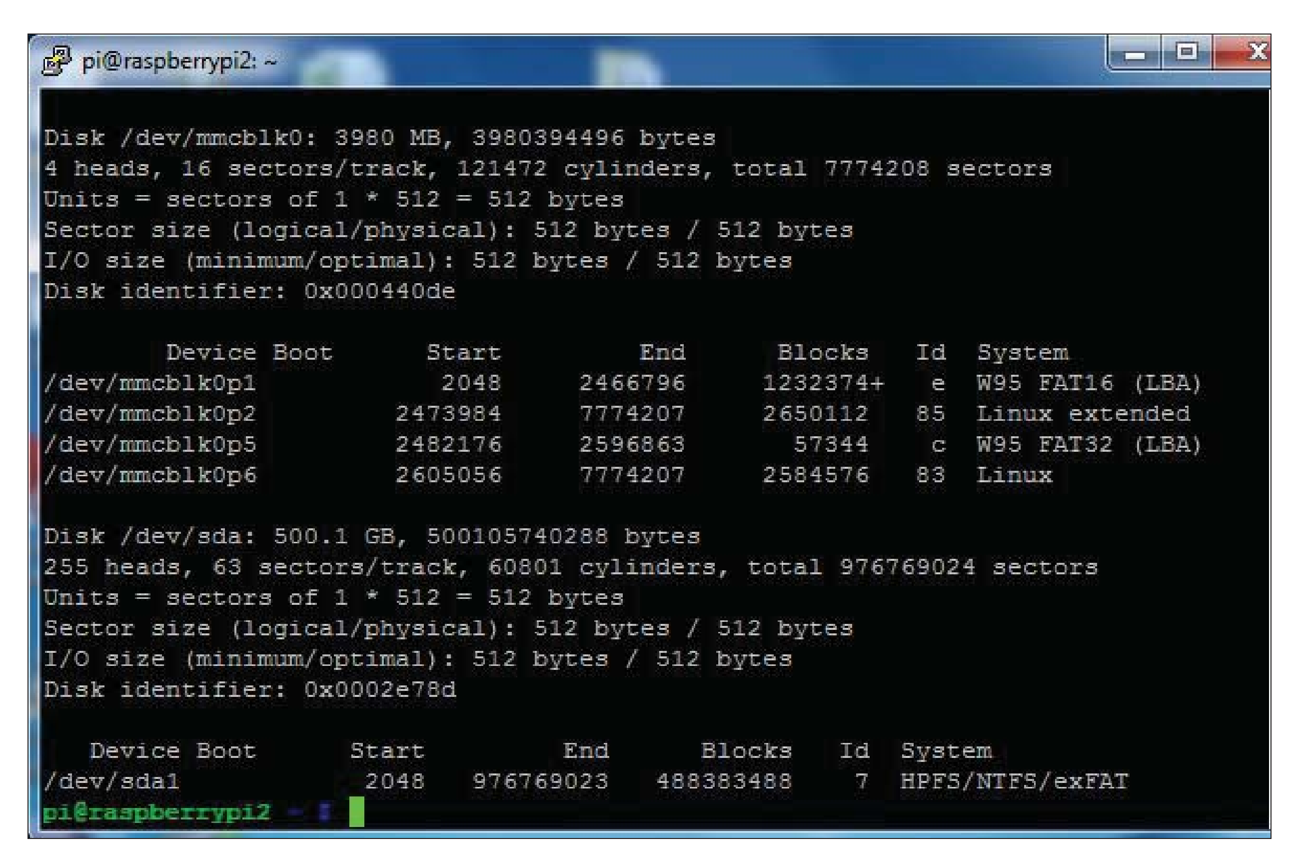


Рис. 3.17. Список разделов

Теперь создаем каталог для монтирования:

```
sudo mkdir /mnt/Arhiv
```

И предоставляем доступ к нему всем группам:

```
sudo chmod 777 /mnt/Arhiv
```

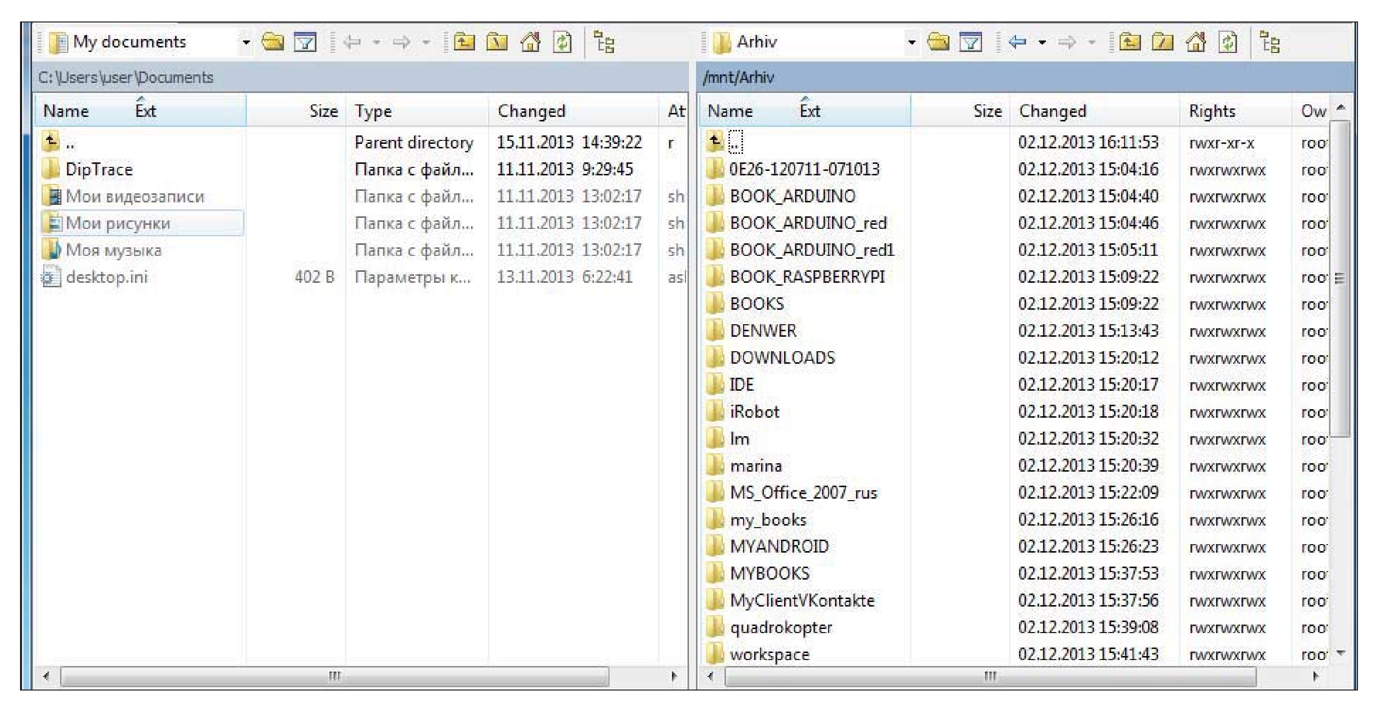


Рис. 3.18. Результат монтирования диска

Сделаем так, чтобы наш диск монтировался при загрузке. Открываем файл /etc/fstab:

```
sudo nano /etc/fstab
```

и добавляем в конец файла следующую строку:

```
/dev/sda1 /mnt/Arhiv ntfs-3g defaults,rw 0 1
```

Перезагружаем систему и видим, что диск примонтирован (рис. 3.18).

3.8. Подключение Wi-Fi

Применений Raspberry Pi можно придумать великое множество, и не всегда при этом удобно вести к нему сетевой кабель. Решим эту проблему с помощью адаптера Wi-Fi, подключаемого через USB. У меня в наличии имелся адаптер, купленный в Китае по цене 7 долларов (рис. 3.19). Позиционируется он в интернет-магазине как устройство специально для Raspberry Pi.

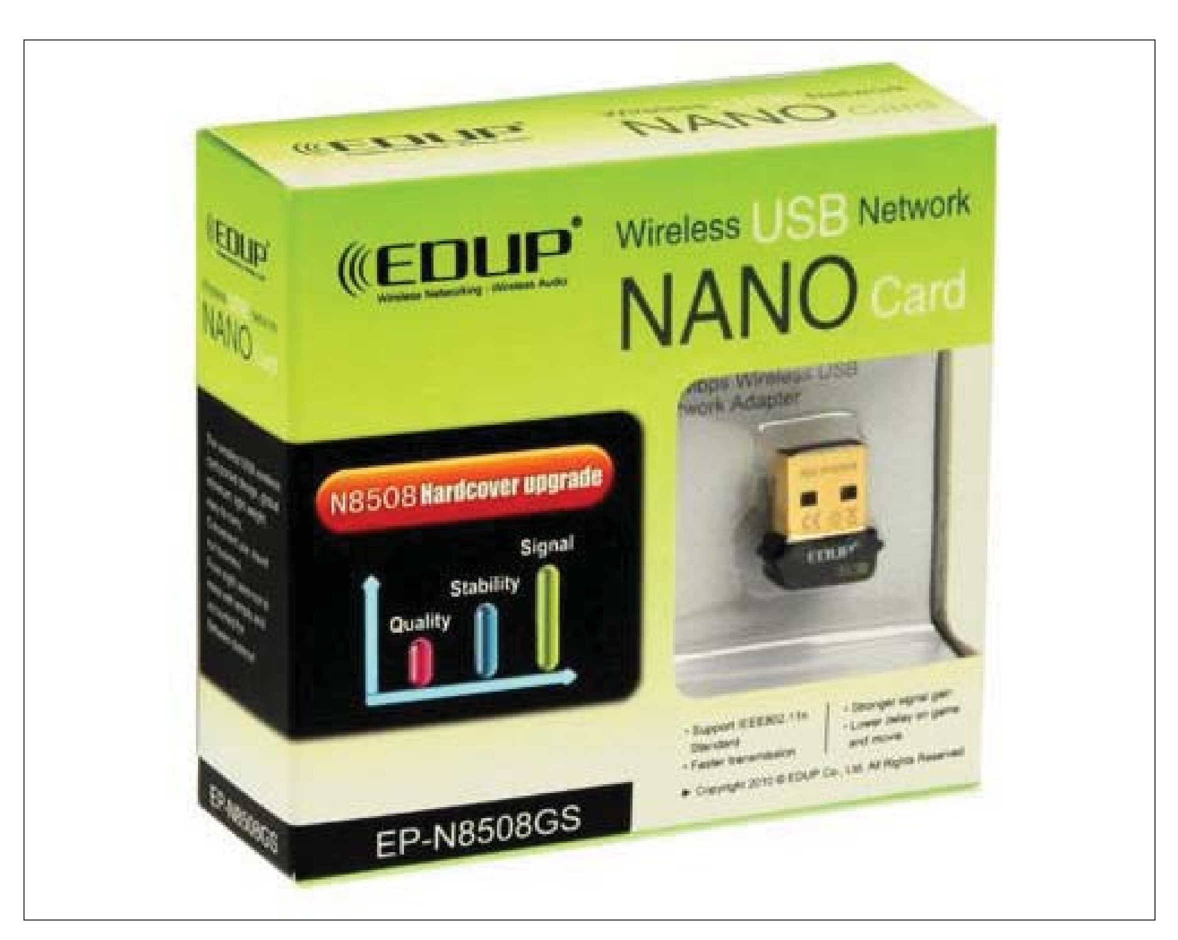


Рис. 3.19. USB-адаптер Wi-Fi

Еще раз напомню, что при подключении адаптера Wi-Fi, как и любого другого внешнего устройства, запитывать Raspberry Pi следует от достаточно мощного блока питания.

Итак, подключаем адаптер в USB-порт и проверяем, определит ли его система. Результат выполнения команды `iwconfig` (рис. 3.20) показывает, что наше устройство найдено.

```
pi@raspberrypi2: ~
Файл  Правка  Вид  Поиск  Терминал  Справка
pi@raspberrypi2 ~ $ iwconfig
wlan0     unassociated  Nickname:"<WIFI@REALTEK>"
          Mode:Managed  Frequency=2.412 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

lo        no wireless extensions.

eth0      no wireless extensions.

pi@raspberrypi2 ~ $
```

Рис. 3.20. Результат выполнения команды `iwconfig`

Теперь сканируем пространство на поиск доступных беспроводных сетей командой:

```
sudo iwlist wlan0 scan
```

и видим (рис. 3.21), что найдена точка доступа **AndroidAP** (в данном случае это планшет Samsung GalaxyTab 2, настроенный как точка доступа Wi-Fi).

```
pi@raspberrypi2: ~
Файл  Правка  Вид  Поиск  Терминал  Справка
pi@raspberrypi2 ~ $ sudo iwlist wlan0 scan
wlan0     Scan completed :
          Cell 01 - Address: 9C:E6:E7:F9:56:79
                    ESSID:"AndroidAP"
                    Protocol:IEEE 802.11bgn
                    Mode:Master
                    Frequency:2.437 GHz (Channel 6)
                    Encryption key:on
                    Bit Rates:65 Mb/s
                    Extra:rsn_ie=30140100000fac040100000fac040100000fac020c00
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : CCMP
                        Pairwise Ciphers (1) : CCMP
                        Authentication Suites (1) : PSK
                    Quality=100/100  Signal level=94/100

pi@raspberrypi2 ~ $
```

Рис. 3.21. Поиск доступных беспроводных сетей

Теперь соединимся с точкой доступа по WPA-шифрованию. Соединение с таким шифрованием поддерживает утилита `wpa_supplicant`.

С помощью утилиты `wpa_passphrase`, которая входит в состав пакета `wpa_supplicant`, генерируем пароль на основе ключа доступа (для моей точки доступа AndroidAP — lcwt9634):

```
wpa_passphrase AndroidAP lcwt9634
```

Утилита выдает сгенерированную строку **psk** (рис. 3.22).

```
pi@raspberrypi2: ~
Файл Правка Вид Поиск Терминал Справка
pi@raspberrypi2 ~ $ wpa_passphrase AndroidAP lcwt9634
network={
        ssid="AndroidAP"
        #psk="lcwt9634"
        psk=cbca148e9a54e30be4b478634809483ce4552289aff6e529bc6d8bf1b26fb63d
}
pi@raspberrypi2 ~ $ sudo nano -w /etc/wpa_supplicant/wpa_supplicant.conf
pi@raspberrypi2 ~ $
```

Рис. 3.22. Генерирование строки **psk**

Далее блок, содержащий ssid сети и строку psk, вставляем в конец конфигурационного файла /etc/wpa_supplicant/wpa_supplicant.conf (рис. 3.23).

```
pi@raspberrypi2: ~
Файл Правка Вид Поиск Терминал Справка
  GNU nano 2.2.6             File: /etc/wpa_supplicant/wpa_supplicant.conf

ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
ssid="AndroidAP"
psk=cbca148e9a54e30be4b478634809483ce4552289aff6e529bc6d8bf1b26fb63d
}
```

Рис. 3.23. Добавление данных в конфигурационный файл wpa_supplicant.conf

В завершение перезагружаем наш Raspberry Pi и наблюдаем подключение к точке доступа на планшете Samsung (рис. 3.24).

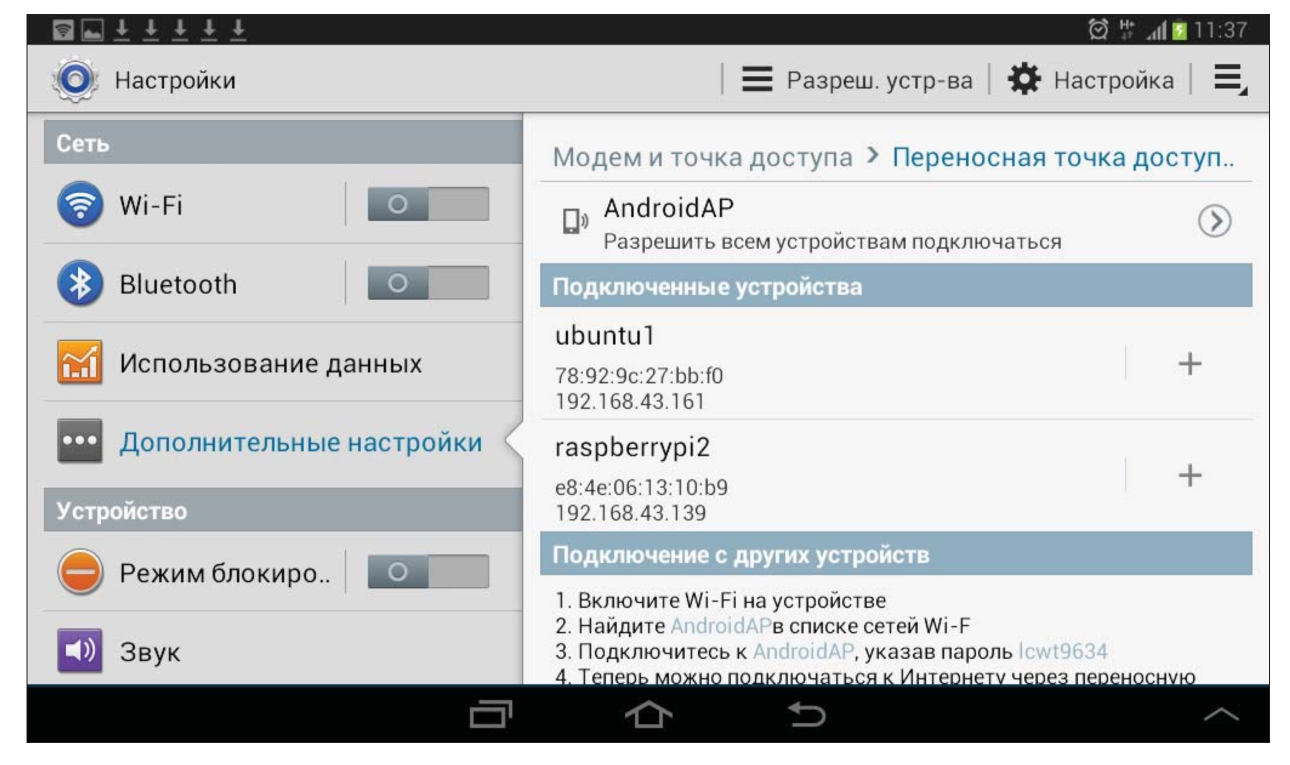


Рис. 3.24. Подключение к точке доступа Wi-Fi

3.9. Подключение 3G-модема

Поскольку сотовая связь доступна практически на всей населенной территории РФ, бесспорное достоинство 3G-модема состоит в том, что в любой точке зоны охвата можно соединиться с Интернетом, пусть и на относительно малых скоростях. Особенно это ощущается в поездках, когда просто нет других доступных способов выхода в Сеть. Очень удобны 3G-модемы USB, имеющие малый размер и доступный порт соединения USB (рис. 3.25).



Рис. 3.25. 3G-модем ZTE MF180 от Билайн

Приобретая USB 3G-модем для подключения к Rapberry Pi, сначала желательно проверить наличие данной модели в списке совместимого оборудования (**http://elinux.org/RPi_VerifiedPeripherals**). Мой модем в этом списке отсутствовал, однако подключить его и получить Интернет на скорости 3G у меня получилось.

Итак, для подключения USB 3G-модема прежде всего необходимо установить пакеты `ppp` и `sakis3g`. Для установки пакета `ppp` выполним команду:

```
sudo apt-get install ppp
```

Затем скачиваем пакет `sakis3g`:

```
sudo wget "http://www.sakis3g.org/versions/latest/armv4t/sakis3g.gz"
```

Этот источник очень часто бывает недоступен, и попав в такую ситуацию, можно обратиться к альтернативному:

```
sudo wget "http://raspberry-at-home.com/files/sakis3g.tar.gz"
```

Далее распаковываем пакет в предварительно созданную папку /usr/bin/modem3g:

```
sudo mkdir /usr/bin/modem3g
sudo chmod 777 /usr/bin/modem3g
sudo cp sakis3g.tar.gz /usr/bin/modem3g
cd /usr/bin/modem3g
sudo tar -zxvf sakis3g.tar.gz
sudo chmod +x sakis3g
```

Теперь можно запустить скрипт `sakis3g`, работающий в терминале:

```
sudo ./sakis3g –interactive
```

В открывшемся главном меню программы (рис. 3.26) выбираем подключение к сети 3G. Скрипт `sakis3g` считывает с модема настройки и предлагает выбор точки доступа для подключения (рис. 3.27).

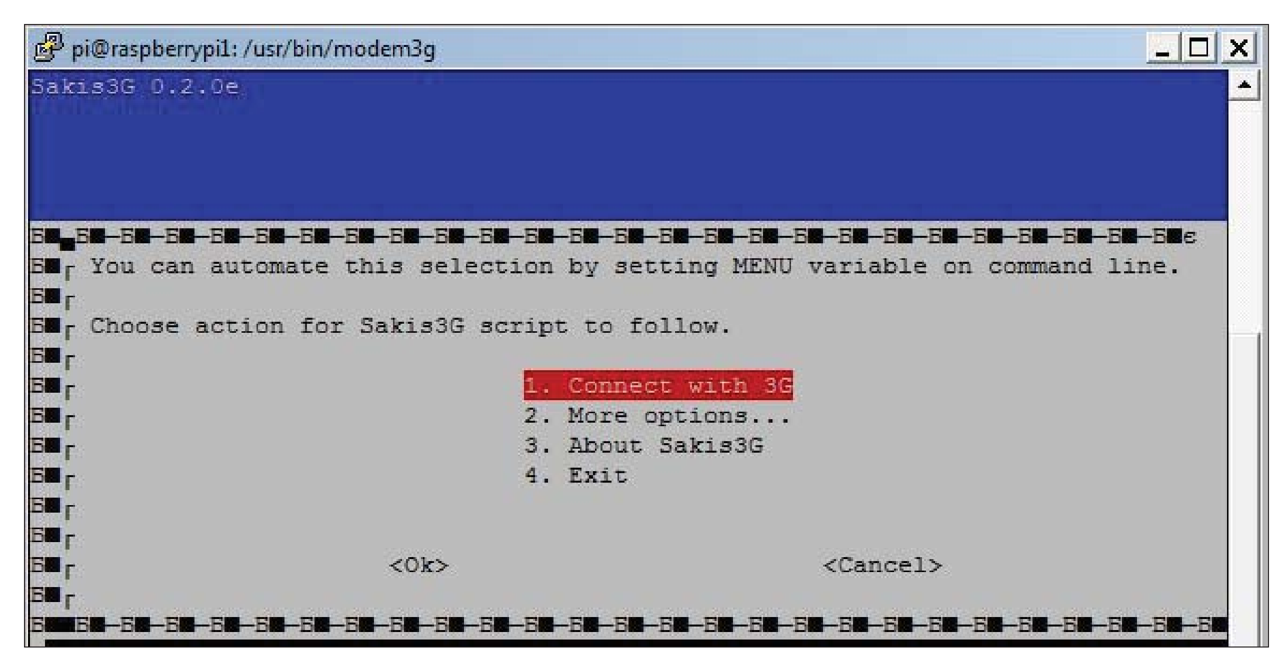


Рис. 3.26. Главное меню программы **sakis3g**

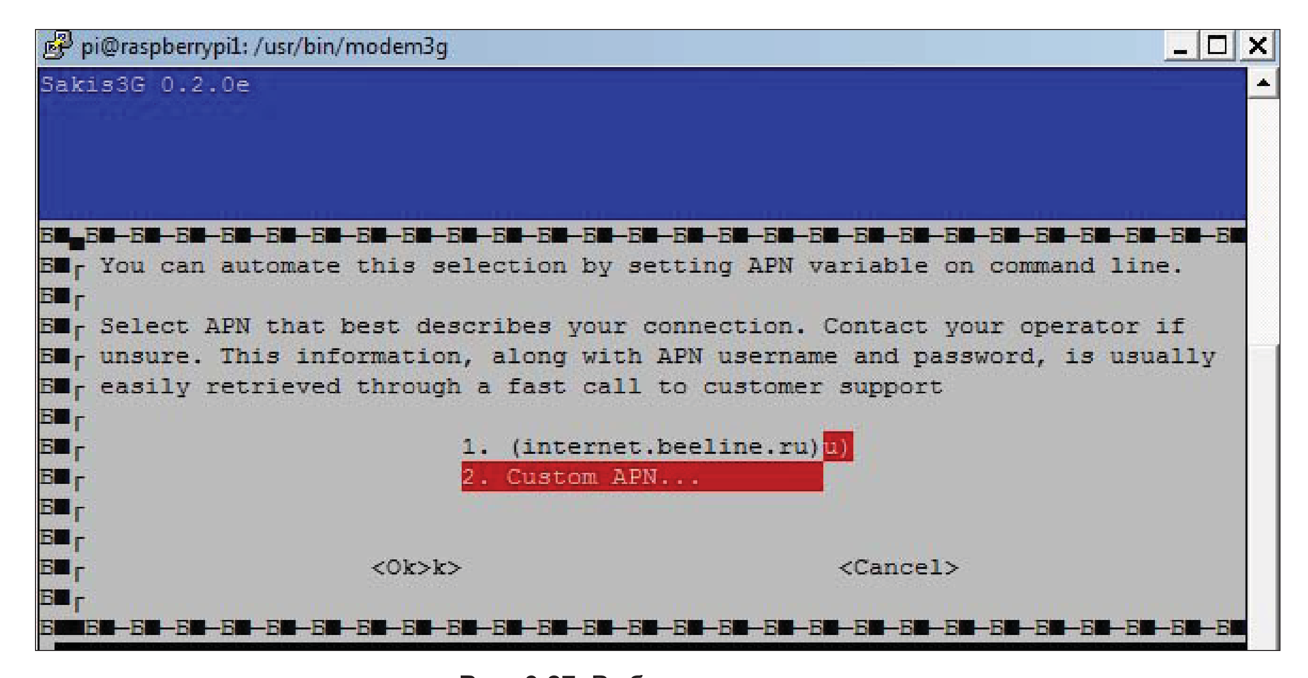


Рис. 3.27. Выбор точки доступа

Для моего тарифа приходится вводить свои параметры точки доступа: `home.beeline.ru` (рис. 3.28).

Затем вводим логин (рис. 3.29) и пароль. Скрипт пытается подключиться к сети и в случае успеха выдает сообщение о подключении (рис. 3.30) или об ошибке подключения.

Далее мы попадаем в меню (рис. 3.31), откуда можно посмотреть настройки соединения (рис. 3.32) или разорвать соединение.

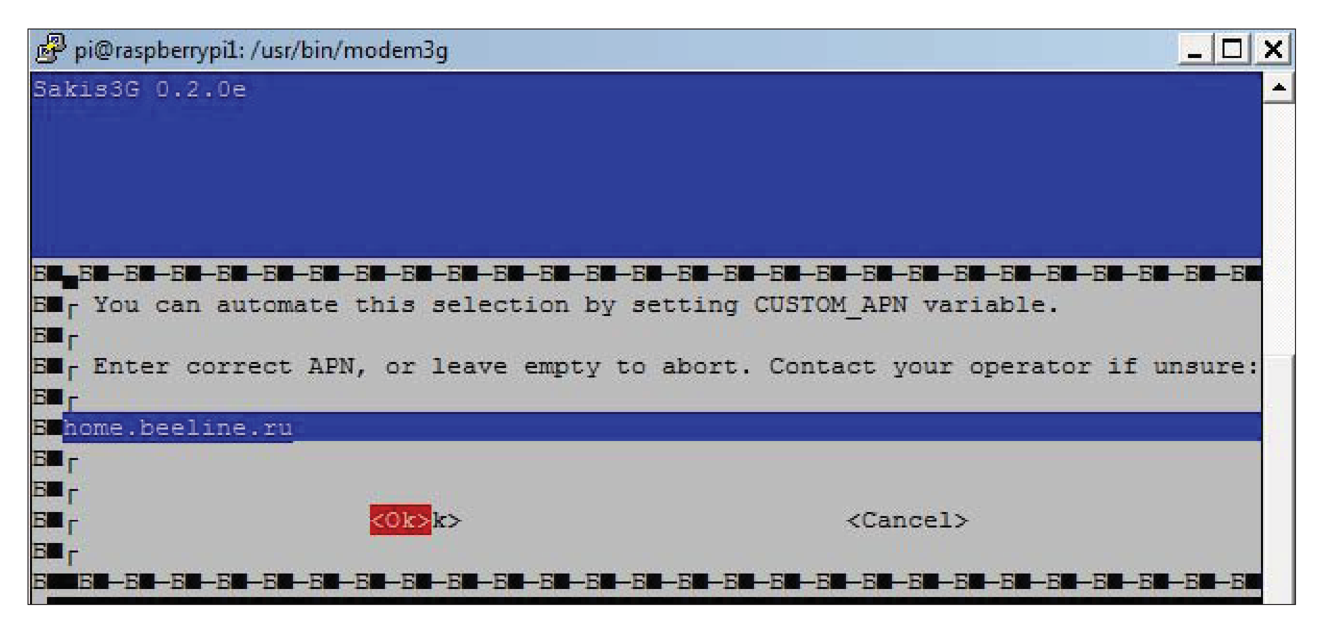


Рис. 3.28. Ввод параметров точки доступа

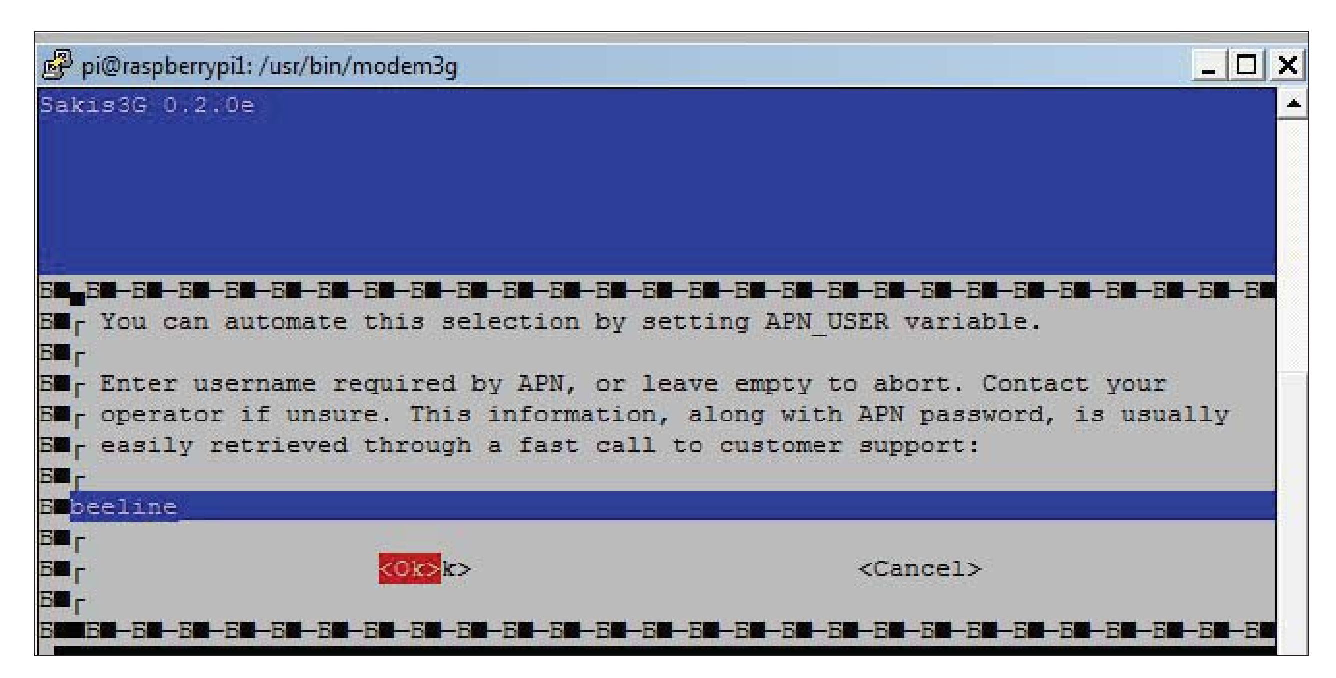


Рис. 3.29. Ввод логина/пароля для точки доступа



Рис. 3.30. Сообщение о подключении к точке доступа

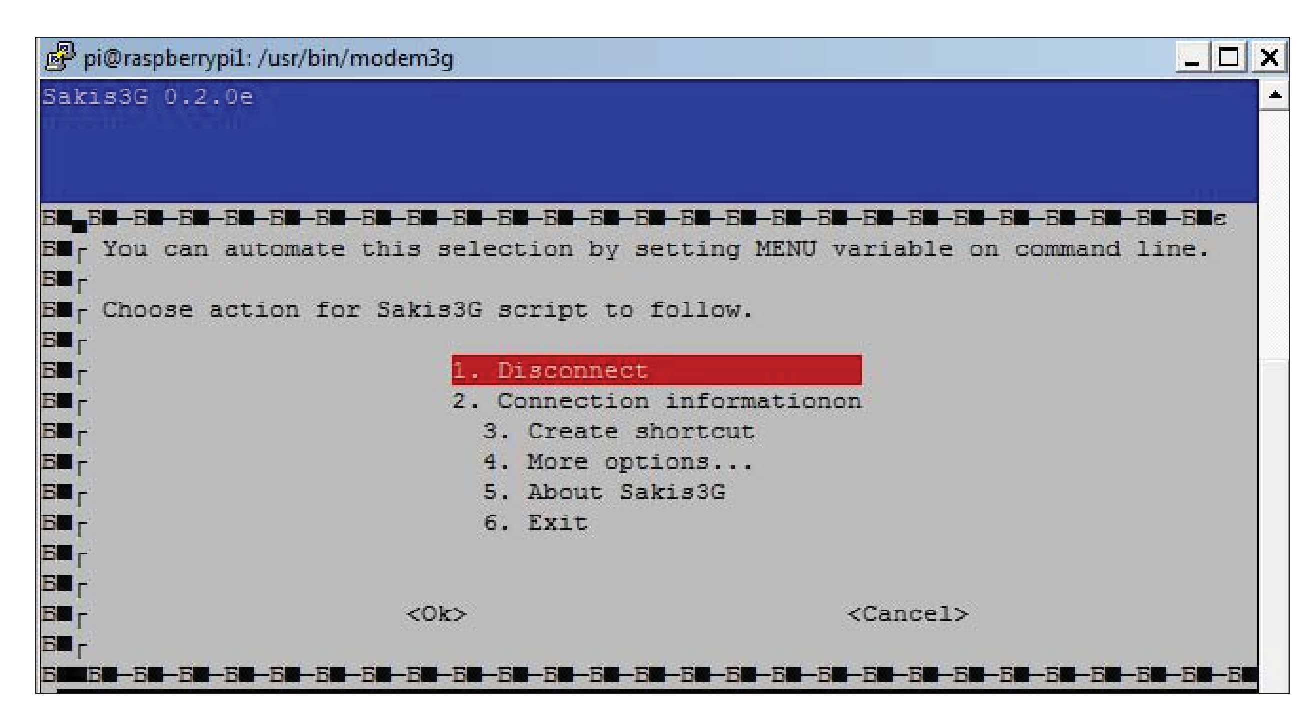


Рис. 3.31. Меню программы **sakis3g**

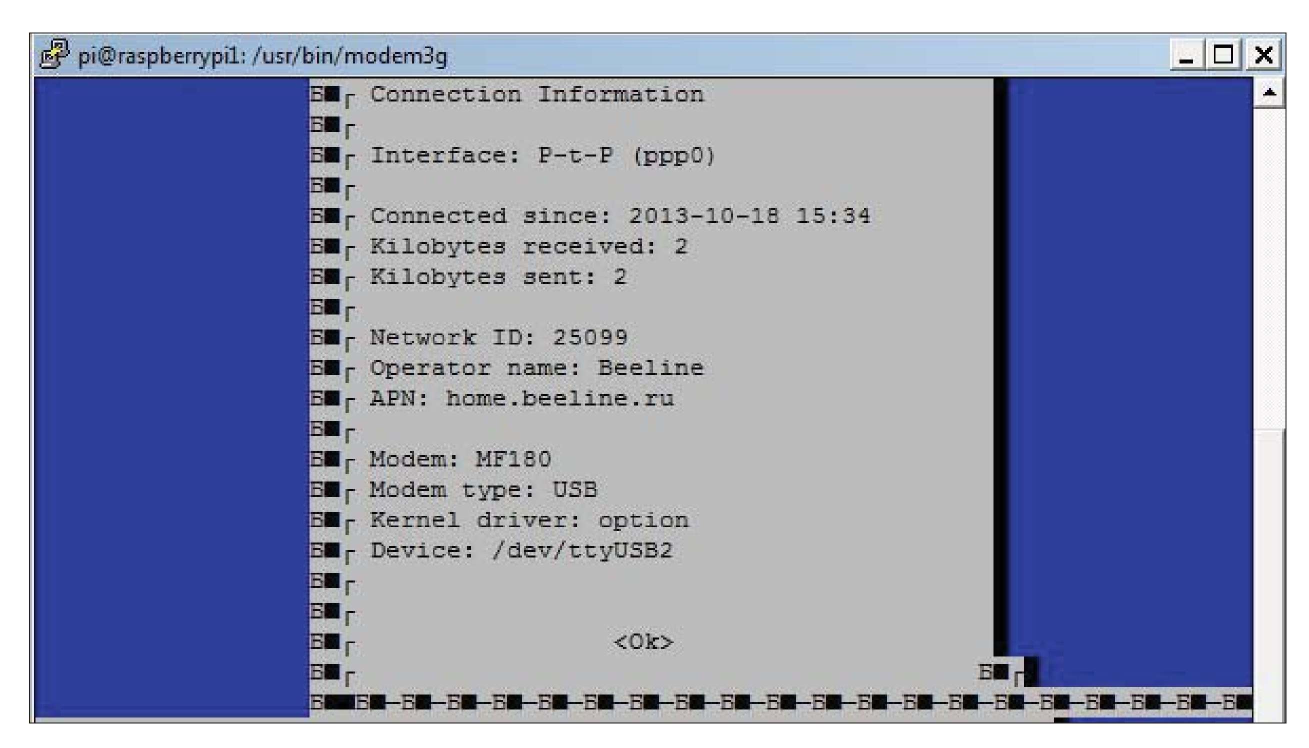


Рис. 3.32. Просмотр данных соединения

3.10. Подключение веб-камеры USB

При подключении веб-камеры USB Raspberry Pi ее обнаруживает сразу же. Увидеть это можно в списке подключенных устройств:

```
lsusb
```

Или в списке устройств видео:

```
ls /dev/video*
```

```
pi@raspberrypi ~ $ ls /dev/video*
/dev/video0  /dev/video1
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 2341:0043 Arduino SA Uno R3 (CDC ACM)
Bus 001 Device 012: ID 1a40:0101 Terminus Technology Inc. 4-Port HUB
Bus 001 Device 013: ID 1220:0008
Bus 001 Device 014: ID 046d:0825 Logitech, Inc. Webcam C270
Bus 001 Device 015: ID 0bda:8176 Realtek Semiconductor Corp. RTL8188CUS 802.11n
WLAN Adapter
Bus 001 Device 017: ID 18ec:3399 Arkmicro Technologies Inc.
pi@raspberrypi ~ $
```

Рис. 3.33. Просмотр подключенных к системе веб-камер USB

```
pi@raspberrypi ~ $ v4l2-ctl --all --device=/dev/video1
Driver Info (not using libv4l2):
        Driver name   : uvcvideo
        Card type     : USB2.0 PC CAMERA
        Bus info      : usb-bcm2708_usb-1.3.4
        Driver version: 3.10.25
        Capabilities  : 0x84000001
                Video Capture
                Streaming
                Device Capabilities
        Device Caps   : 0x04000001
                Video Capture
                Streaming
Priority: 2
Video input : 0 (Camera 1: ok)
Format Video Capture:
        Width/Height  : 640/480
        Pixel Format  : 'YUYV'
        Field         : None
        Bytes per Line: 1280
        Size Image    : 614400
        Colorspace    : Unknown (00000000)
Crop Capability Video Capture:
        Bounds      : Left 0, Top 0, Width 640, Height 480
        Default     : Left 0, Top 0, Width 640, Height 480
        Pixel Aspect: 1/1
Streaming Parameters Video Capture:
        Capabilities     : timeperframe
        Frames per second: 30.000 (30/1)
        Read buffers     : 0
                     brightness (int)    : min=0 max=255 step=1 default=-8193 va
lue=110
                       contrast (int)    : min=0 max=255 step=1 default=57343 va
lue=150
                     saturation (int)    : min=0 max=255 step=1 default=57343 va
lue=60
                            hue (int)    : min=-127 max=127 step=1 default=-8193
 value=0
                          gamma (int)    : min=1 max=8 step=1 default=57343 valu
e=4
                           gain (int)    : min=0 max=65535 step=1 default=57343
value=16
           power_line_frequency (menu)   : min=0 max=2 default=1 value=1
                      sharpness (int)    : min=0 max=255 step=1 default=57343 va
lue=1
pi@raspberrypi ~ $
```

Рис. 3.34. Просмотр параметров камеры

На рис. 3.33 показано, что система определила две камеры (подключены Logitech C270 и Defender Glory 32). Камеры включены в USB-хаб с внешним источником питания (см. рис. 3.6).

Для настройки параметров камеры устанавливаем приложение `v4l2-ctl`:

```
sudo apt-get install v4l-utils
```

Для просмотра списка параметров подключенных камер (рис. 3.34) выполняем команды:

```
v4l2-ctl –all –device=/dev/video0
v4l2-ctl –all –device=/dev/video1
```

Для интерактивной установки параметров камеры можно воспользоваться приложением с графическим интерфейсом `luvcview`:

```
sudo apt-get install luvcview
```

Запускаем установленное приложение:

```
luvcview -s 640x480 -i 30
```

Меняя разрешение и частоту кадров, можно подобрать оптимальные значения, при которых камера будет давать наиболее быстрый поток.

3.11. Подключение камеры Raspberry Camera Board

Для Raspberry Pi специально разработана камера Raspberry Camera Board (рис. 3.35). Самое главное отличие этой камеры от прочих USB-камер — возможность подключения ее напрямую к графическому процессору через разъем CSi на плате, что позволяет записывать и кодировать в h.264 изображение с камеры без использования процессорного времени.

Технические характеристики камеры Raspberry Camera Board:

- матрица — Omnivision 5647 с фиксированным фокусом, 5 мегапикселов;
- поддержка разрешений 1080p30, 720p60, 640×480 p60/90 при записи видео;
- возможность делать снимки с разрешением 2592×1944 пикселов;
- размеры платы: 25×20×9 мм;
- изображение с камеры может быть продублировано на HDMI-выход (так называемый режим preview).

В комплект камеры включен ZIF-шлейф. Разъем для подключения шлейфа на плате находится между портами Ethernet и HDMI (рис. 3.36).

Для включения поддержки камеры в Raspbian запускаем программу **raspi-config**:

```
sudo raspi-config
```

и в пункте **Enable Camera** выбираем опцию **Enable** (рис. 3.37). Перезагружаемся.



Рис. 3.35. Камера Raspberry Camera Board



Рис. 3.36. Подключение шлейфа Raspberry Camera Board

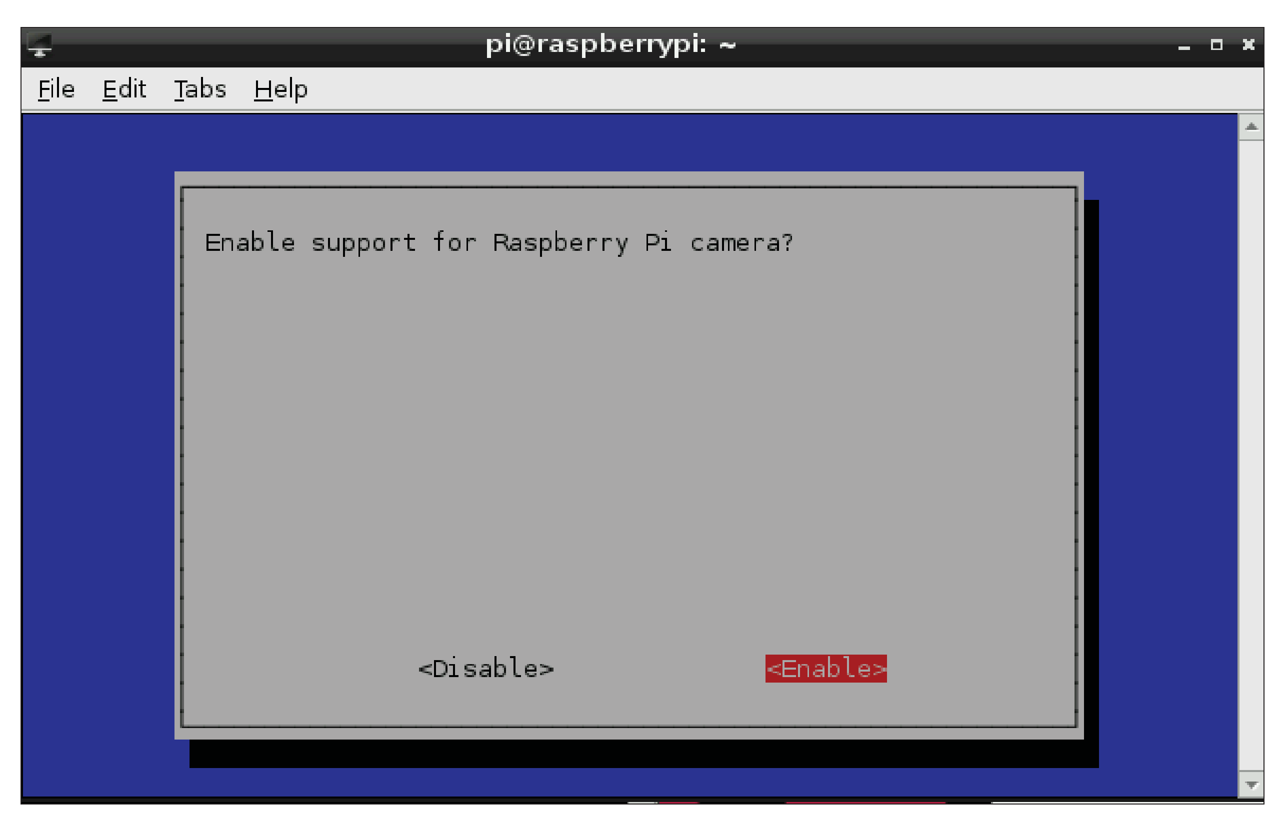


Рис. 3.37. Включение поддержки камеры Raspberry Camera Board в меню **raspi-config**

Для работы с камерой Raspberry Camera Board используются консольные приложения: `raspivid` — записи видео и `raspistill` — для получения изображений.

Параметры этих приложений:

- `-o` или `-output` — определяют имя выходного файла;
- `-t` или `-timeout` — задают длительность записи видео (по умолчанию 5 секунд);
- `-d` или `-demo` — запись в режиме демонстрации возможностей, будут использованы все возможные эффекты.

Примеры:

- `raspistill -o image.jpg` — захват изображения в формате JPG;
- `raspivid -o video.h264` — захват 5-секундного видео в формате h.264;
- `raspivid -o video.h264 -t 10000` — захват 10-секундного видео в формате h264.

Посмотреть все возможные параметры для приложений `raspivid` и `raspistill` можно следующим образом:

```
raspivid | less
raspistill | less
```

3.12. Неисправности Raspberry Pi и борьба с ними

Рассмотрим основные неисправности и проблемы, возникающие при работе с Raspberry Pi, и некоторые способы их решения.

3.12.1. Проблемы с питанием или в момент включения

Качественный источник питания — самый главный залог отсутствия проблем. Рассмотрим основные проблемы.

Красный индикатор не горит, нет изображения на экране

Отсутствует питание устройства. Необходимо проверить источник питания.

Красный индикатор мигает

Если красный светодиод мигает при подключении к источнику питания, значит, напряжение питания менее 5 В. В этой ситуации нужно заменить источник питания.

Красный индикатор горит, зеленый не мигает, нет изображения на экране

Это может означать, что Raspberry Pi не сумел обнаружить образ операционной системы на SD-карте. Проверьте, что карта установлена правильно. Убедитесь, что на карте памяти правильно записан образ операционной системы. Подключите SD-карту к компьютеру под управлением Windows и убедитесь в наличии на карте файлов bootcode.bin, loader.bin, start.elf и прочих. Также проверьте, что карта памяти совместима. Проверьте контакты разъема SD-карты — они должны хорошо пружинить и выступать не менее чем на 2 мм от нижнего края держателя.

Попробуйте отключить от Raspberry Pi все кабели, за исключением кабеля питания. Вставьте SD-карту и включите компьютер. Зеленый индикатор **OK** должен мигать в течение примерно 20 секунд. Если это помогло, то необходимо подключать остальные кабели по одному, чтобы определить, какой из них мешает загрузке.

Возможно напряжение питания недостаточно (менее 5 В). Попробуйте заменить источник питания или его кабель. Как уже отмечалось, Raspberry Pi требует блок питания с током не менее 700 мА и более.

Зеленый индикатор мигает в определенном порядке

На свежих версиях прошивок зеленый индикатор мигает в определенном порядке, в зависимости от неисправности:

- 3 вспышки — файл loader.bin не найден;
- 4 вспышки — файл loader.bin не загружается;
- 5 вспышек — файл start.elf не найден;
- 6 вспышек — файл start.elf не загружается.

Если файл start.elf не может загрузиться, то, возможно, он поврежден и может быть заменен файлом arm240_start.elf или другим из файлов типа arm*XXX*_start.elf.

На экране появляется только разноцветный квадрат

В свежих прошивках после загрузки самой прошивки (файл start.elf) отображается разноцветный квадрат. Если возникает такой квадрат и загрузка останавливается,

это означает, что не получилось загрузить файл kernel.img. Попробуйте заменить этот файл исправным.

Кроме того, сразу после появления такого квадрата у Raspberry Pi несколько возрастает потребление тока питания. Поэтому, если устройство перезагружается в этот момент, можно заподозрить, что блок питания не способен обеспечить необходимую мощность. Из-за недостатка мощности снижается напряжение, что и приводит к перезапуску.

Ошибка *Kernel Panic* при загрузке

Загрузка начинается, но затем останавливается и выдается соответствующее сообщение. Чаще всего, такая ошибка связана с несовместимыми устройствами USB — как правило, с клавиатурами. Попробуйте отключить клавиатуру и загрузить устройство снова.

Raspberry Pi выключается или перезагружается сразу после загрузки

Причина такого поведения заключается в недостаточном напряжении питания.

Компьютер иногда загружается, но не каждый раз

При гарантированно исправном блоке питания и проверенной SD-карте Raspberry Pi загружается лишь время от времени. В остальных случаях зеленый индикатор **OK** загорается, но загрузки не происходит даже при отключенных устройствах USB и отключенном сетевом подключении. Виноваты в этом либо блок питания, либо SD-карта. Некоторые устройства нормально работают лишь до тех пор, пока не нагреются (блоки питания и SD-карты также подвержены нагреву). Проверьте блок питания, а также контакты держателя SD-карт.

3.12.2. Клавиатура, мышь и другие устройства ввода

Компьютер не реагирует на клавиатуру, или нажатая клавиша многократно повторяется

Все эти проблемы возникают из-за недостатка питания. Необходимо использовать мощный блок питания с качественным кабелем. Некоторые блоки питания с некачественными проводами дают достаточно энергии для зарядки сотового телефона, но недостаточно для питания Raspberry Pi. Ряд устройств USB требуют очень высокой мощности источника питания. На большинство таких устройств нанесена маркировка с указанием требуемого напряжения питания и тока. Минимально они потребляют 5 В 100 мА каждое, поэтому чаще всего их приходится подключать через отдельный USB-хаб (концентратор) с внешним питанием.

Попробуйте отключить все устройства USB, за исключением клавиатуры. Обратите внимание на то, что некоторые клавиатуры имеют встроенный USB-концентратор, и тогда они потребляют порядка 150 мА, в то время как сам Raspberry Pi предос-

тавляет лишь 100 мА на каждый порт USB при отсутствии дополнительного питания.

Клавиатура и мышь не работают вместе с USB-адаптером Wi-Fi

Одновременное подключение клавиатуры или мыши и USB-адаптера Wi-Fi может вызвать в работе всех устройств сбои, возникающие из-за большой разницы в скорости обмена высокоскоростного адаптера Wi-Fi и низкоскоростных клавиатуры и мыши. Тем не менее, многие подобные проблемы решаются подбором более качественного блока питания. Возможно, стоит попробовать разные варианты подключения — например, адаптер Wi-Fi подключать непосредственно к Raspberry Pi, а клавиатуру с мышью — к USB-концентратору с внешним питанием. Имеет смысл попробовать и другие варианты, но в любом случае нужно помнить, что качественное питание устройству необходимо в первую очередь.

Проблемы с беспроводной клавиатурой

Некоторые беспроводные клавиатуры — например, Microsoft Wireless Keyboard 800 — не работают с Raspberry Pi, несмотря даже на достаточное питание. Причиной этого является ее несовместимость с драйвером операционной системы. Перед покупкой клавиатуры убедитесь, что она входит в список совместимого оборудования.

Введенные символы не соответствуют клавиатуре

Если введенные символы на экране не соответствуют указанным на клавиатуре, нужно изменить раскладку клавиатуры. Для этого введите следующую команду:

```
sudo dpkg-reconfigure keyboard-configuration
```

Далее следуйте указаниям на экране. После выхода из программы перезагрузите компьютер.

Поменять раскладку клавиатуры можно и через конфигурационное меню **raspi-config**, вызываемое командой:

```
sudo raspi-config
```

Долго загружаются настройки клавиатуры

Если вы заменили раскладку клавиатуры, после чего время загрузки компьютера увеличилось, попробуйте ввести следующую команду:

```
sudo setupcon
```

3.12.3. Обновление прошивки Raspberry Pi

Использование свежей версии прошивки устройства может решить разные проблемы, в том числе и связанные, например, с несовместимостью SD-карт или мониторов.

Для обновления прошивки выполните команду:

```
sudo apt-get update
```

Проверить версию ядра Linux можно следующей командой:

```
uname -a
```

Прошивка графического сопроцессора GPU проверяется так:

```
/opt/vc/bin/vcgencmd version
```

Прошивку графического сопроцессора можно обновить при помощи утилиты Hexxeh's rpi-update tool.

Существует несколько моделей разделения памяти между процессором ARM и графическим сопроцессором GPU:

- arm240_start.elf — 240M ARM, 16M GPU split (максимум памяти процессора, подходит для задач, в которых не используются видео и 3D);
- arm224_start.elf — 224M ARM, 32M GPU split (режим минимально достаточный для запуска графической среды и простых программ);
- arm192_start.elf — 192M ARM, 64M GPU split (средний режим для простого видео или 3D, используется по умолчанию);
- arm128_start.elf — 128M ARM, 128M GPU split (используйте этот режим для серьезных задач 3D или просмотра видео вместе с 3D, он также необходим для XBMC).

Для переключения вручную необходимо заменить файл start.elf одним из здесь перечисленных и перезагрузить компьютер. Например:

```
sudo cp /boot/arm240_start.elf /boot/start.elf && sudo reboot
```

3.12.4. SD-карты

При возникновении проблем сначала убедитесь, что установлена последняя версия прошивки. Некоторые SD-карты не совместимы с Raspberry Pi, поэтому необходимо проверить их на совместимость. В случае любых проблем с SD-картой начать стоит с форматирования карты, особенно если на карте уже были какие-то данные.

После записи образа системы на SD-карту необходимо, подключив карту памяти к компьютеру, проверить, что на ней существует загрузочный раздел boot. Этот раздел должен содержать несколько файлов, в том числе — обязательно — start.elf и kernel.img. Если этих файлов на карте памяти нет, то, очевидно, проблема возникает при записи на карту.

Необходимо убедиться, что в момент записи и после него SD-карта не защищена от записи. Многие карты имеют на корпусе переключатель, который запрещает производить на нее запись. Кроме того, из-за возможных неисправностей некоторые кардридеры неверно определяют положение этого переключателя и ошибочно считают, что запись на такие карты памяти запрещена. В этом случае можно попробовать заменить кардридер или карту памяти. Кстати, большинство проблем с карта-

ми памяти чаще всего связаны с неисправными кардридерами. И если кардридер неверно определяет состояние защиты от записи, иногда проблема снимается установкой переключателя защиты от записи в промежуточное положение. Тогда кардридер считывает его состояние верно.

3.12.5. Звук

Нет звука на мониторе, подключенном по HDMI

Это происходит потому, что некоторые компьютерные мониторы используют режим DVI даже при подключенном кабеле HDMI. В случае, когда другие устройства, подключенные к этому монитору по HDMI, работают нормально, можно исправить проблему с Raspberry Pi, добавив в конфигурационный файл config.txt следующую строку:

```
hdmi_drive=2
```

Это заставит монитор переключиться в режим HDMI.

Нет звука совсем или в отдельных приложениях

В Raspbian звук отключен по умолчанию, т. к. звуковой драйвер ALSA до сих пор находится в состоянии альфа-версии (т. е. не протестирован и не отлажен). Для того чтобы включить звук, до запуска графической оболочки командой `startx` необходимо ввести следующие команды:

```
sudo apt-get install alsa-utils
sudo modprobe snd_bcm2835
```

В свежих версиях Raspbian (Debian Wheezy) драйвер `snd_bm2835` включен изначально, поэтому указанный шаг не нужен. Можно попробовать так:

```
sudo aplay /usr/share/sounds/alsa/Front_Center.wav
```

По умолчанию выход звука определяется автоматически (т. е. HDMI, если устройство способно проигрывать звуки, в остальных случаях — выход наушников). Выход звука можно задать вручную:

```
sudo amixer cset numid=3 <n>
```

где `n = 0` — автоматический режим, `n = 1` — наушники, `n = 2` — HDMI.

Для проверки наличия звука можно использовать программу `hello_audio`. В свежих выпусках прошивок Raspberry Pi для компиляции программы введите следующие команды:

```
cd /opt/vc/src/hello_pi/
./rebuild.sh
cd hello_audio
```

После этого можно запустить программу для проверки аналогового выхода (наушников):

```
./hello_audio.bin
```

или для проверки HDMI:

```
./hello_audio.bin 1
```

Также может понадобиться добавить в группу audio пользователя, под учетными данными которого вы запускаете программу, для получения прав на доступ к звуковому устройству.

3.12.6. Изображение

Команда *startx* не выполняется

Ошибки при запуске графической среды командой `startx` могут возникать из-за недостатка места на SD-карте. По умолчанию на карте размером 2 Гбайт остается всего несколько сотен мегабайт свободного места. Это место быстро занимается пользовательскими файлами. Необходимо убедиться, что на карте памяти есть свободное место, или расширить раздел до максимального размера, если у вас карта памяти объемом больше 2 Гбайт.

Также при установке некоторые программы могут повредить файл настроек .Xauthority, который находится в домашнем каталоге. Для устранения проблемы надо переименовать, переместить в другой каталог или удалить этот файл.

Неверные цвета на экране

Если монитор подключен по кабелю DVI — надо проверить, что разъем надежно зафиксирован винтами.

Видео не воспроизводится или воспроизводится очень медленно

Поддержка аппаратного ускорения видео реализована лишь в проигрывателе XBMC и в проигрывателе для командной строки `omxplayer`. Аппаратное ускорение поддерживается только для показа видеофайлов, закодированных h.264. Аппаратное кодирование видео не поддерживается.

Большие черные поля вокруг небольшого изображения на мониторе высокой четкости (HD)

При первом запуске изображение может не заполнять экран полностью. Этот эффект называется "сжатая развертка" и легко исправляется. Необходимо добавить следующие строки в конфигурационный файл config.txt:

```
disable_overscan=1
```

Для увеличения размера изображения:

```
overscan_left=-20
overscan_right=-20
overscan_top=-20
overscan_bottom=-20
```

Точная регулировка размера изображения производится экспериментальным подбором параметров в файле config.txt.

Изображение выходит за границы экрана

При первом запуске Raspberry Pi может выдавать на телевизоре в режиме 1080p изображение, выходящее за границы экрана. Этот эффект называется overscan и может быть исправлен изменениями в конфигурационном файле config.txt. Делается это по аналогии с предыдущим пунктом, только использовать следует положительные числа — например:

```
overscan_left=20
overscan_right=20
overscan_top=20
overscan_bottom=20
```

Помехи или искажение цветов на мониторах HDMI или DVI

Чаще всего это может быть вызвано помехами в некачественном или слишком длинном видеокабеле.

3.12.7. Проблемы с сетью

Соединение теряется при подключении устройства USB

Чаще всего это происходит из-за недостаточной мощности источника питания. Используйте качественный источник питания и качественные провода. Некоторые устройства USB потребляют большой ток (более 100 мА), поэтому должны подключаться с помощью концентратора (хаба) с внешним питанием. Кроме того, некоторые дешевые USB-концентраторы потребляют ток от Raspberry Pi даже при наличии внешнего источника питания.

Бывают также случаи, когда сетевое соединение теряется при совместном подключении через концентратор с внешним питанием низкоскоростных устройств USB — например, мыши или клавиатуры. В такой ситуации можно посоветовать подключить эти устройства напрямую к USB-портам Raspberry Pi (конечно, при условии, что ток потребления каждого из них не превышает 100 мА).

Микросхемы сетевого адаптера и контроллера USB сильно греются

Это нормально. На открытом воздухе и при комнатной температуре (24 градуса) микросхема LAN9512 контроллера Ethernet/USB может в процессе работы нагреваться до 52 градусов. При такой температуре уже не получается дотрагиваться до ее корпуса более чем на пару секунд, но это абсолютно нормальный режим работы для этой микросхемы.

Сеть перестает работать при переносе SD-карты с одного Raspberry Pi на другой

В некоторых дистрибутивах в файле /etc/udev/rules.d/70-persistent-net.rules запоминается, какой MAC-адрес привязан к интерфейсу eth0, поэтому изменение MAC-адреса

приводит к созданию нового интерфейса (eth1, eth2 и т. д.). Необходимо отредактировать файл /etc/udev/rules.d/70-persistent-net.rules, убрав оттуда ненужные привязки, и перезагрузить компьютер.

Происходят сбои при высокой нагрузке на сеть

Драйверу USB выделяется оперативная память из области ядра системы, поэтому при очень большой нагрузке на сеть (например, при использовании торрентов) нехватка памяти вызывает критические сбои или зависания системы. Найдите в файле /etc/sysctl.conf строку:

```
vm.min_free_kbytes = 8192
```

Пробуйте увеличить параметр до `16384` или большего значения. Если это не поможет, попробуйте добавить в файл /boot/cmdline.txt строку:

```
smsc95xx.turbo_mode=N
```

Это снизит скорость сетевого обмена, но позволит избежать сбоев.

Пропадает сетевое соединение при запуске графической среды

Сетевое соединение может пропадать при запуске графической среды командой `startx`. Это вызвано ошибкой в драйвере USB, связанной с отдельными типами USB-мышей.

3.12.8. Проблемы с GPIO

Всегда надо помнить, что логические уровни сигнала шины GPIO составляют 3,3 В, и использование сигналов с уровнем 5 В недопустимо. Если в результате кратковременного замыкания двух контактов разъема GPIO между собой или замыкания контакта питания на землю ваш Raspberry Pi перестал подавать признаки жизни — не пугайтесь. Это последствия срабатывания защитных предохранителей. Они автоматически восстанавливаются через некоторое время после устранения причины замыкания (требуется время, чтобы предохранитель остыл и полимер, содержащийся в нем, кристаллизовался). На восстановление предохранителей может понадобиться несколько часов, поэтому лучше отложить устройство в сторону и подождать.

Контакты разъема GPIO подключены напрямую к процессору ARM и весьма чувствительны к статике, поэтому постарайтесь не касаться этих контактов во время работы. И вообще, касаясь выводов GPIO или оборудования, подключенного к GPIO, всегда используйте заземление.



ГЛАВА 4

Дистрибутив Raspbian — настройка и установка дополнительных пакетов

4.1. Поддержка русского языка

Для поддержки русского языка в консоли набираем в терминале следующую команду:

```
sudo apt-get install console-cyrillic
sudo dpkg-reconfigure -plow console-cyrillic
```

В окне настроек последовательно выбираем: клавиши переключения языков, в качестве кодировки — UNICODE, подходящий шрифт, устанавливаем настройку кириллицы при загрузке системы. Сохраняем. Поддержка русского языка в консоли установлена.

4.2. Файловый менеджер

В качестве файлового менеджера можно использовать Midnight Commander (рис. 4.1). Для его установки в терминале выполните команду:

```
sudo apt-get install mc
```

Запустите менеджер:

```
mc
```

4.3. Создание скриншотов

Очень часто необходимо получить скриншот экрана. Для этого можно использовать пакет `imagemagick` или, лучше, еще более легкий `scrot`.

Установка:

```
sudo apt-get install scrot
```

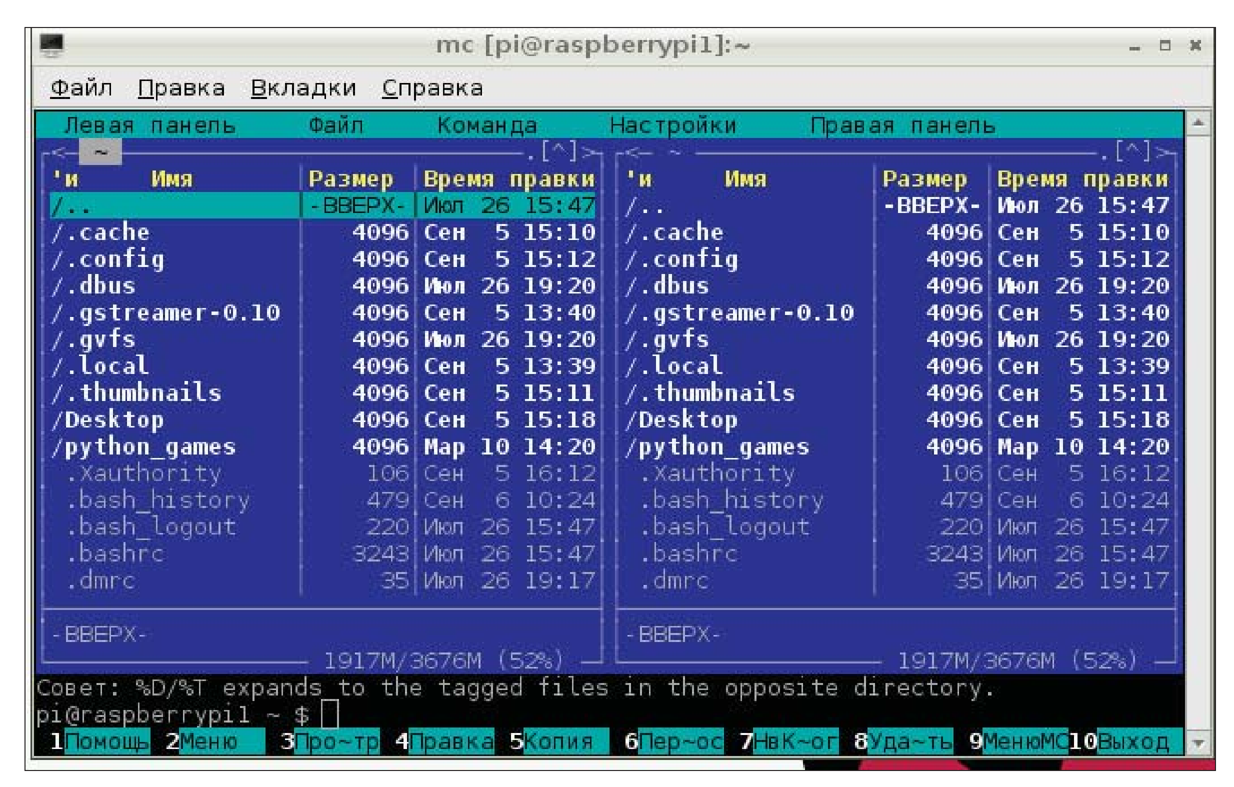


Рис. 4.1. Файловый менеджер Midnight Commander

Для получения скриншота набираем в терминале:

```
scrot
```

Для получения скриншота через 5 секунд:

```
scrot -d 5
```

4.4. Доступ к Raspberry Pi по SSH в консольном и графическом режиме

В большинстве проектов Raspberry Pi используется не в качестве полноценного настольного ПК, а как сетевой мини-компьютер или устройство для управления различной электроникой. Для подобной работы Raspberry Pi не требуются ни монитор, ни клавиатура. Гораздо удобнее воспользоваться в этом случае монитором и клавиатурой основного компьютера и работать с Raspberry Pi по локальной сети. Такое взаимодействие двух компьютеров реализуется с помощью специального сетевого протокола SSH. Для включения SSH на Raspberry Pi зайдем в конфигурационное меню **raspi-config** и включим поддержку SSH при загрузке в пункте **Advanced Options | SSH** (рис. 4.2).

Далее узнаем IP-адрес нашего Raspberry Pi и подключаемся к нему с любого компьютера сети с помощью SSH-клиента. С Windows-компьютеров это можно сделать с помощью программы PuTTY (об этом чуть далее), а с компьютеров под Linux набираем в терминале:

```
ssh pi@192.168.0.10
```

и получаем доступ по SSH к консоли Raspberry Pi (рис. 4.3).

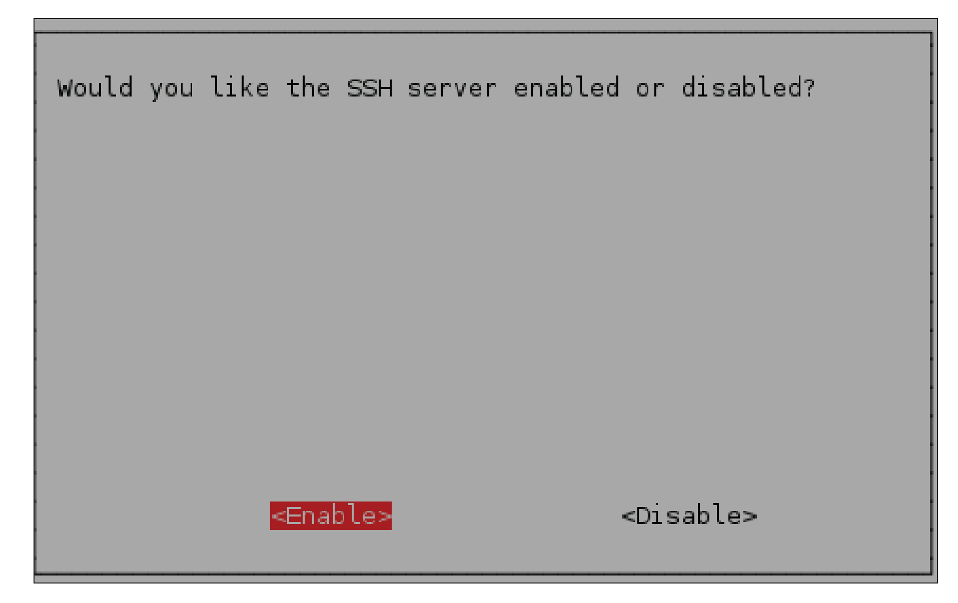


Рис. 4.2. Активация поддержки SSH в Raspberry Pi

```
pi@raspberrypi1: ~
login as: pi
pi@192.168.0.10's password:
Linux raspberrypi1 3.6.11+ #474 PREEMPT Thu Jun 13 17:14:42 BST 2013 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Sep  6 10:47:46 2013
pi@raspberrypi1 ~ $ ^C
pi@raspberrypi1 ~ $
```

Рис. 4.3. Доступ к консоли Raspberry Pi по SSH

Если мы хотим на Linux-клиенте запускать по SSH на Raspberry Pi графические приложения, то выполняем команду `ssh` с ключом `X`:

```
ssh -X pi@192.168.0.10
```

Ключ `X` заставляет перенаправлять данные графического интерфейса от Raspberry Pi (192.168.0.10) на главный ПК через SSH-соединение (так называемый X-Forwarding). После установки соединения мы получаем доступ к консоли Raspberry Pi, но теперь при запуске любого графического приложения Raspberry (например,

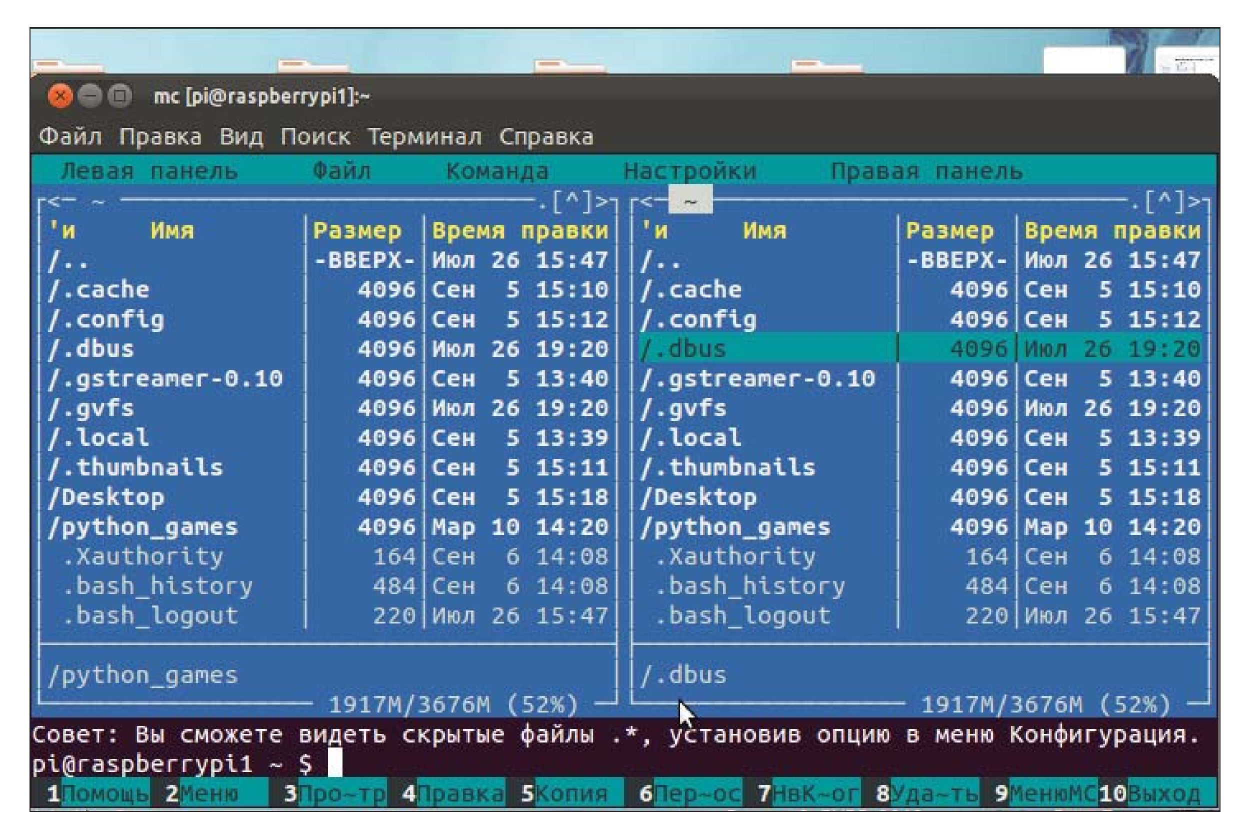


Рис. 4.4. Запуск графических изображений по SSH

файлового менеджера mc) его окно откроется прямо на нашем рабочем столе (рис. 4.4).

При запуске в SSH-клиенте команды `startlxde` осуществляется трансляция рабочего стола Raspberry Pi в окно Linux-клиента. Удаленный рабочий стол Raspbian останется полностью функциональным, но с некоторыми ошибками: рабочий стол и значки окажутся от Raspberry, а панель задач — от главного ПК. При этом переключаться между рабочими столами обеих машин станет невозможно. Такое поведение объясняется конфликтом оконных менеджеров двух компьютеров. Чтобы этого избежать, на главном ПК нужно запустить отдельную независимую сессию X-сервера без оконного менеджера на другом дисплее (под номером 1) и использовать запущенный X-сервер для отображения рабочего стола Raspberry Pi. Для этого на главном ПК набираем в терминале:

```
sudo xinit -- :1
```

Эта команда приведет к запуску на дисплее 1 отдельного X-сервера, в результате чего появится пустой экран с открытой консолью xterm (рис. 4.5).

В этой консоли набираем команду для SSH-соединения с Raspberry Pi и команду запуска графической оболочки:

```
ssh -X pi@192.168.0.10
startlxde
```

Теперь на экране появится удаленный рабочий стол Raspberry Pi, с которым можно полноценно работать так, как будто мы работаем непосредственно с самим мини-компьютером (рис. 4.6).

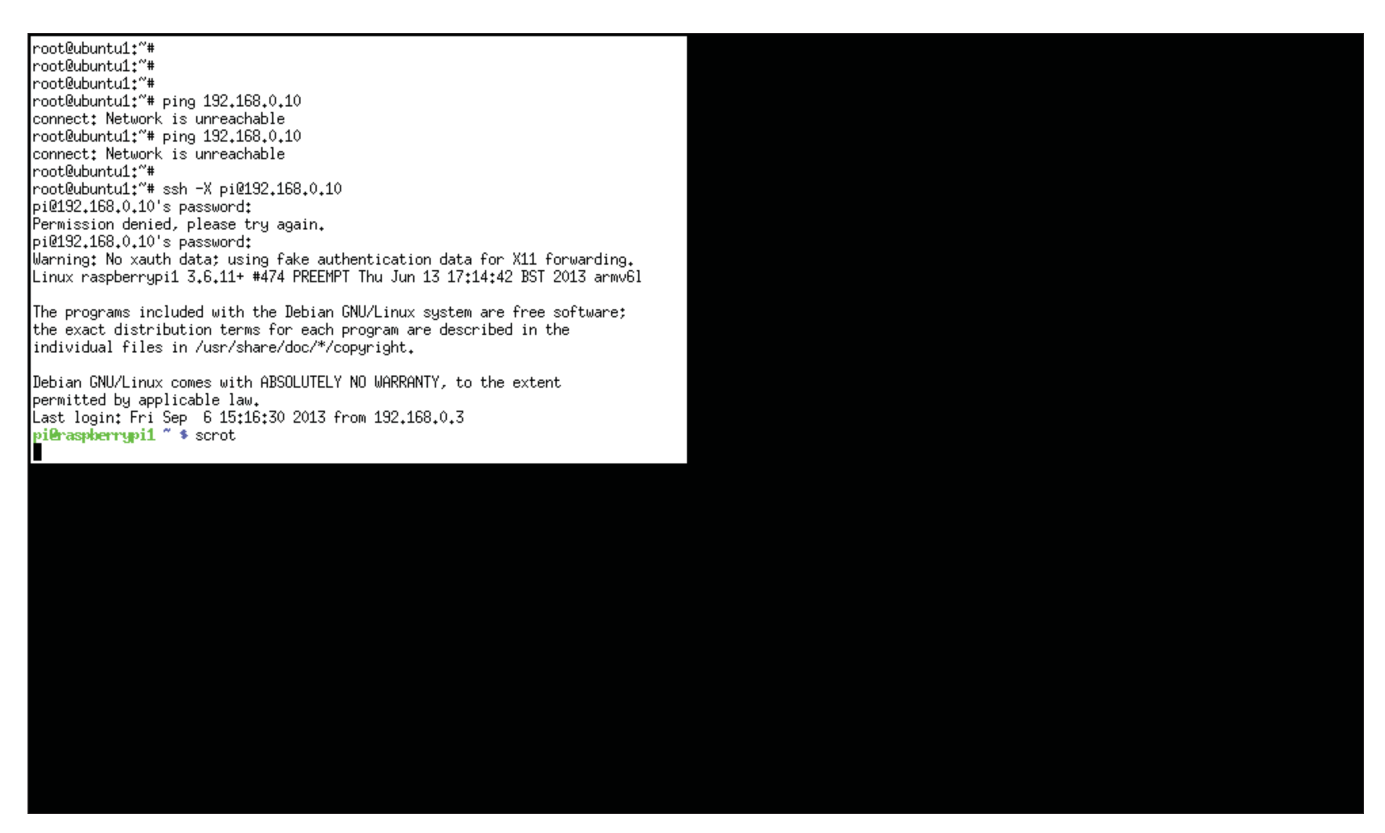


Рис. 4.5. Запуск сервера X на дисплее 1

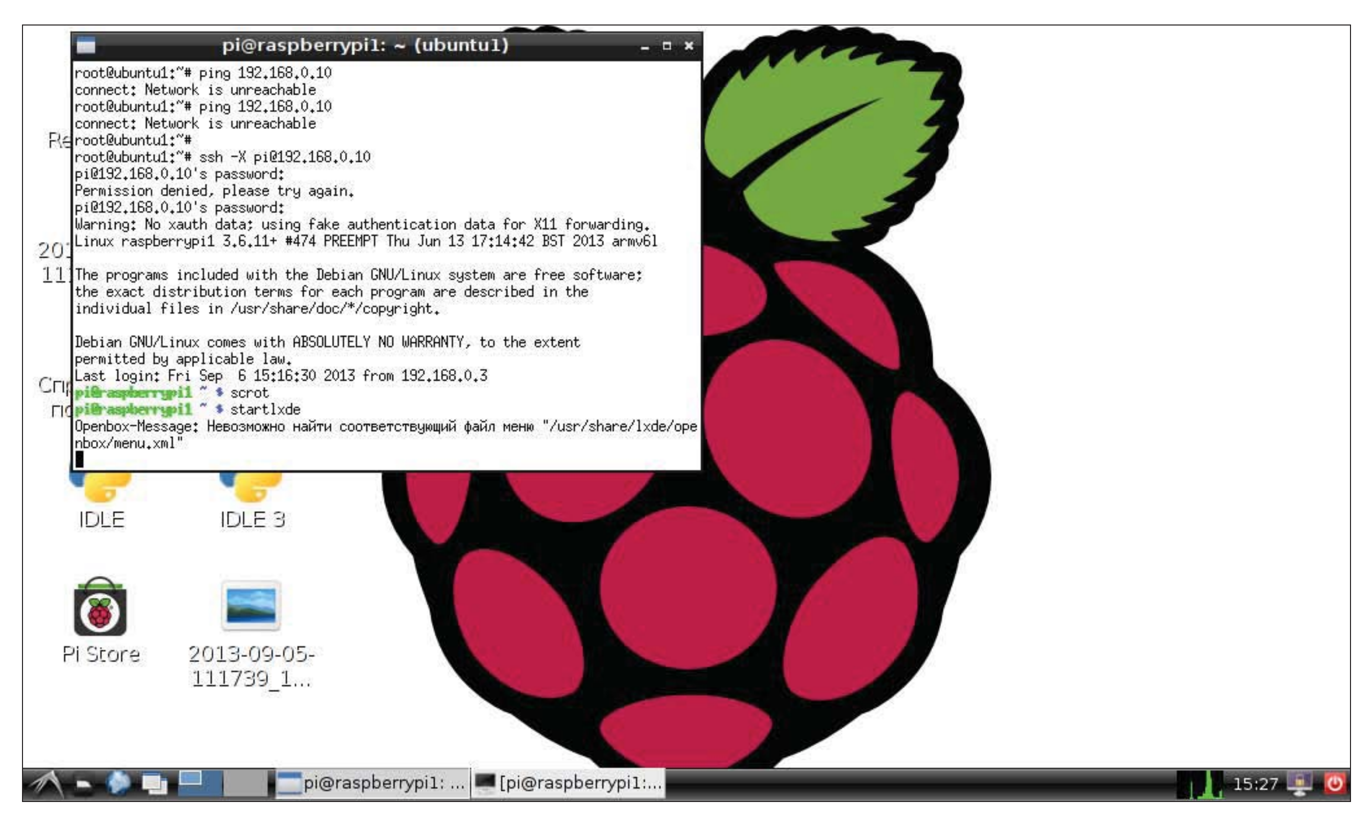


Рис. 4.6. Удаленный рабочий стол Raspberry Pi

Для переключения между дисплеями (рабочим столом главного ПК и удаленным рабочим столом Raspberry Pi) служат сочетания клавиш <Ctrl>+<Alt>+<F8>, <Ctrl>+<Alt>+<F9> (или <Ctrl>+<Alt>+<F7>, <Ctrl>+<Alt>+<F8> в зависимости от дистрибутива Linux). Для завершения сеанса удаленного доступа достаточно закрыть терминал.

Чтобы запускать удаленный рабочий стол Raspbian с Windows-компьютера, вам понадобятся две программы: PuTTY и Xming. PuTTY — это свободно распространяемый SSH-клиент (скачать его можно с сайта **http://www.putty.org/**), а Xming — локальный X-сервер для Windows (бесплатно скачать его можно по адресу **http://sourceforge.net/projects/xming/**). Именно Xming будет отображать рабочий стол на вашем компьютере. Скачав указанные дистрибутивы, установите их на компьютере, откуда вы хотите получить доступ к Raspberry Pi.

В настройках PuTTY выберите в меню пункт **SSH | X11** и установите флажок **Включить переадресацию X11**. А в поле **Отображение дисплея X** впишите `localhost:0`. Соединитесь с Raspberry Pi.

При установке Xming согласитесь со всеми вопросами, которые он вам задаст. При первом запуске на этапе **Select display settings** выберите пункт **One window**, опцию **Display number** установите в **0**. В окне **Select how to start Xming** выберите **Start no client**. Запустите Xming. В области уведомлений панели задач должен появиться его значок, свидетельствующий о том, что X-сервер работает. Соответственно мы можем запустить рабочий стол. Вводим к терминале PuTTY команду: `startlxde` — и перед нами рабочий стол Raspbian.

4.5. Удаленный рабочий стол VNC

Virtual Network Computing (VNC) — система удаленного доступа к рабочему столу компьютера, использующая протокол RFB (Remote FrameBuffer, удаленный кадровый буфер). Управление осуществляется путем передачи нажатий клавиш на клавиатуре и движений мыши с одного компьютера на другой и ретрансляции содержимого экрана через компьютерную сеть.

Система VNC платформонезависима: VNC-клиент, называемый VNC viewer, запущенный на одной операционной системе, может подключаться к VNC-серверу, работающему на любой другой ОС.

Сначала установим на Raspberry Pi сервер VNC. Для этого необходимо выполнить команду:

```
sudo apt-get install tightvncserver
```

Во время установки вам будет задан вопрос об инсталляции программы без проверки подлинности: **Install these packages without verification [y/n]?**. Ответьте на этот вопрос утвердительно: **y**.

После окончания установки выполните команду:

```
vncserver :1 -geometry 1280×800 -depth 16 -pixelformat rgb565
```

Эта команда запустит VNC-сервер (рис. 4.7). В данном случае это означает, что запущена виртуальная X-сессия (виртуальное представление рабочего стола Raspberry Pi), абсолютно аналогично тому, как происходит по команде `startx` при загрузке при подключенном мониторе. Теперь, когда вы запустите клиент VNC и подключитесь к нему, то окажетесь подключены именно к этому виртуальному рабочему столу.

```
pi@raspberrypi1 ~ $ vncserver :1 -geometry 1280.800 -depth 16 -pixelformat rgb56
5
vncserver: geometry 1280.800 is invalid
pi@raspberrypi1 ~ $ vncserver :1 -geometry 1280x800 -depth 16 -pixelformat rgb56
5

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? y
Password:
Verify:

New 'X' desktop is raspberrypi1:1

Creating default startup script /home/pi/.vnc/xstartup
Starting applications specified in /home/pi/.vnc/xstartup
Log file is /home/pi/.vnc/raspberrypi1:1.log

pi@raspberrypi1 ~ $
```

Рис. 4.7. Запуск сервера VNC на Raspberry Pi

Наиболее важная часть команды запуска VNC-сервера — параметр `:1`. Он определяет номер порта, на котором будет запущен процесс VNC. Номер порта может быть любым, но его нужно запомнить, поскольку он понадобится при подключении. Еще один важный параметр — разрешение (в данном случае — размер) виртуального рабочего стола. Его можно задать любым, однако не стоит указывать больше, чем реальное разрешение компьютера, с которого осуществляется удаленный доступ.

При первом запуске VNC-сервер попросит ввести пароль. Этот пароль понадобится при подключении к удаленному рабочему столу. Второй пароль, который запросит сервер VNC, — это пароль "только для просмотра". Если ввести этот пароль при подключении, то видеть виртуальный рабочий стол будет можно, однако клавиатура и мышь окажутся отключены.

Итак, подключаемся к Raspberry Pi с помощью программы UltraVNC[1]. Запускаем UtlraVNC, вводим IP-адрес, номер порта (помните? — `1`) и нажимаем кнопку **Connect** (рис. 4.8). Вводим пароль и попадаем на рабочий стол Raspbian (рис. 4.9).

[1] UltraVNC — это свободное программное обеспечение под операционную систему Microsoft Windows, использующее протокол VNC для управления удаленными рабочими столами на других компьютерах. Скачать его можно по ссылке: **http://www.uvnc.com/downloads.html**.

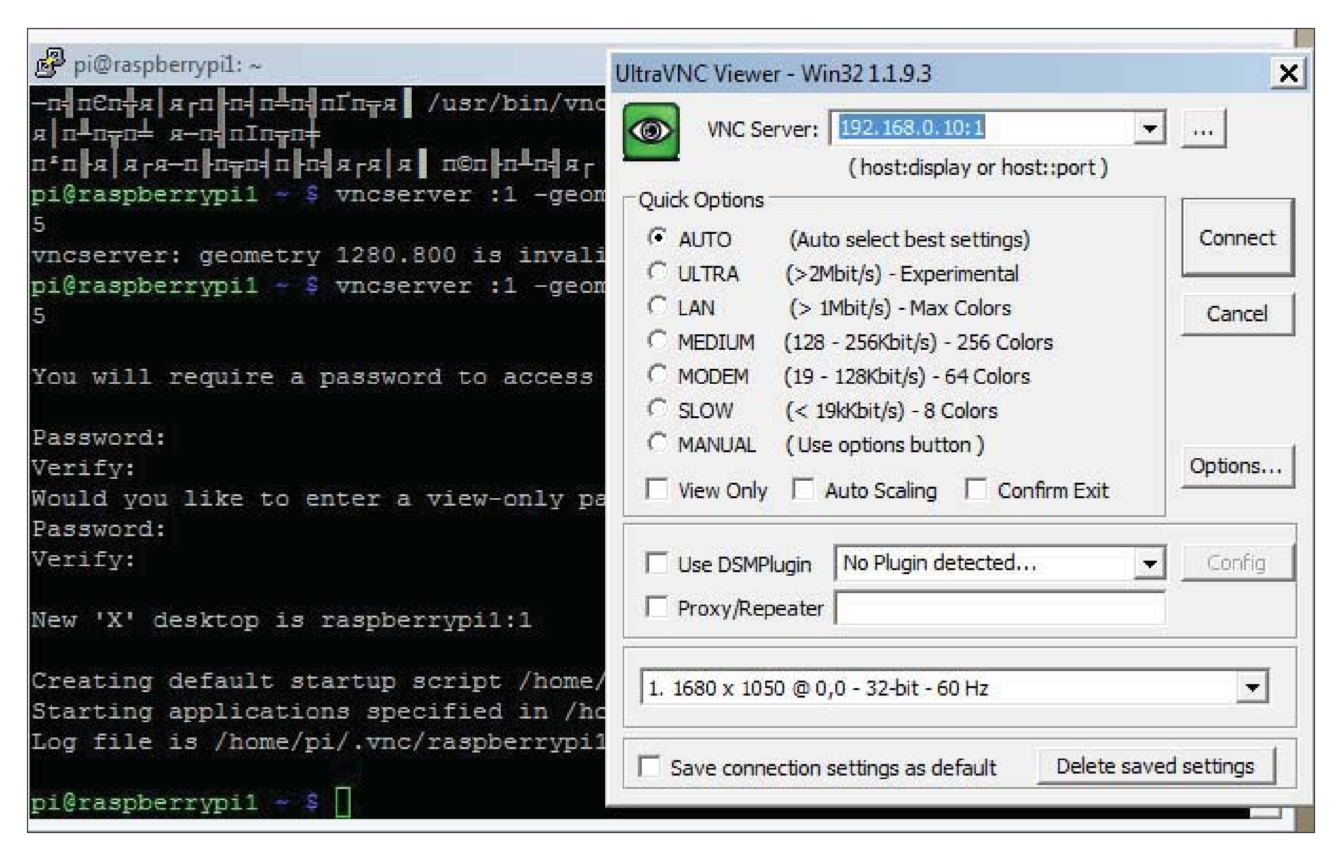


Рис. 4.8. Запуск клиента UltraVNC

Рис. 4.9. Доступ к рабочему столу Raspberry Pi через VNC

4.6. Установка пакета Samba

Пакет Samba представляет собой интерфейс, который обеспечивает связь между UNIX- и Windows-системами в сети, расширяя сетевые возможности ОС UNIX.

Благодаря Samba пользователь, работая на UNIX-системе, может получить доступ к сетевым дискам и принтерам Windows. Для клиентов это выглядит таким образом, будто продолжает работать NT-сервер. Клиенты могут применять сетевое окружение Windows, подсоединять и отсоединять сетевые диски, а также использовать данные на сетевом сервере, не ощущая, работает сервер под UNIX или под Windows. Клиенты могут обращаться к UNIX-файлам, изменять и удалять их (если позволяют права). Таким образом, Samba фактически исполняет функции NT-сервера.

Возможности пакета Samba:

- предоставлять файлы и принтеры Linux/UNIX-сервера для использования под Windows 9*x*/NT/2000/XP и более новых;
- непосредственно управлять пользователями NT;
- оптимально комбинировать безопасность данных и стабильную работу, которые предлагает UNIX-сервер, с операционными системами Microsoft на рабочих станциях;
- поддерживать (не полностью) доменную структуру NT DOMAIN;
- поддерживать функции первичного контроллера NT;
- поддерживать функции участника NT DOMAIN;
- обеспечивать сервисы WINS (клиент и сервер), TIME-server и т. д.;
- придавать UNIX-системе свойства сети NT.

Установим пакет Samba:

```
sudo apt-get install samba samba-common-bin
```

Создадим папку Obmen1:

```
sudo mkdir Obmen1
```

Зададим права для этой папки:

```
chmod -R 777 Obmen1
```

Для настройки Samba открываем файл smb.conf:

```
sudo nano /etc/samba/smb.conf
```

В конец файла дописываем строки:

```
[Flash1]
path = /mnt/flash
writeable = yes
browseable = yes
guest ok = yes
```

```
[Obmen1]
path = /home/pi/Obmen1
writeable = yes
browseable = yes
guest ok = yes
```

Сохраняем файл и перезапускаем Samba:

```
sudo /etc/init.d/samba restart
```

Теперь пробуем с Windows-компьютера по сети открыть наши общие папки (рис. 4.10).

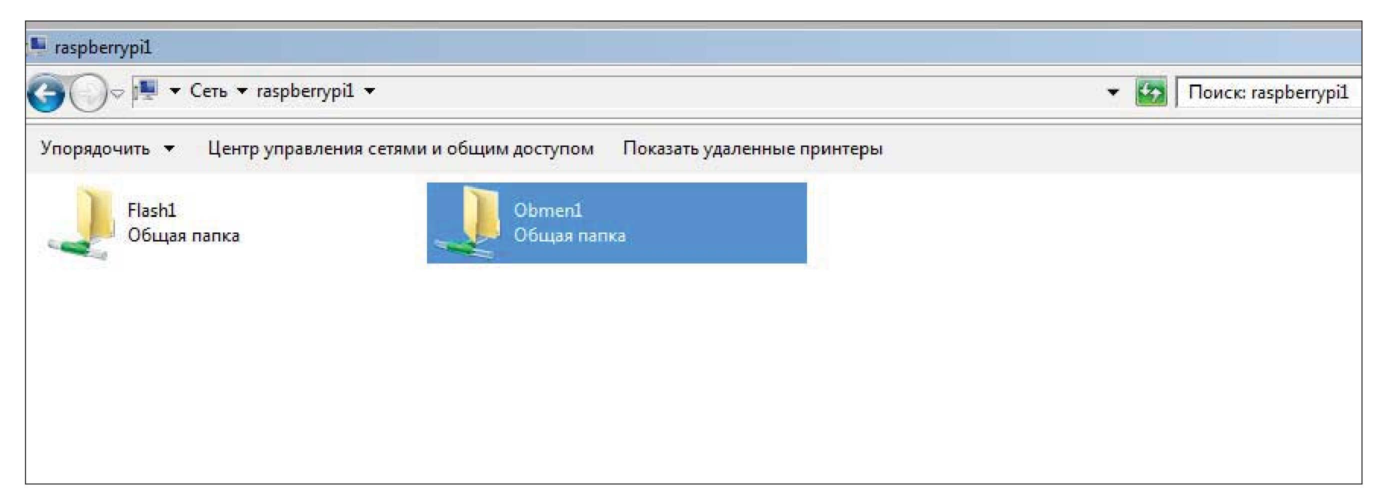


Рис. 4.10. Доступ к общим папкам на Raspberry Pi по сети

Иногда, очень удобно на машине с Linux иметь подключенную (примонтированную) Windows-папку, находящуюся в общем доступе. Аналогично тому, как в самой Windows происходит подключение сетевых папок или дисков, в пакете Samba версии 3.5 имеется приложение mount.cifs, которое поможет нам подмонтировать сетевой раздел.

Итак, создаем на Raspbian папку, куда будем монтировать сетевую Windows-папку:

```
sudo mkdir /mnt/kostyabook
```

И выставляем на нее права:

```
sudo chmod -R 777  /mnt/kostyabook
```

Затем выполняем команду:

```
sudo mount -t cifs //192.168.0.8/BOOKS /mnt/kostyabook -o _netdev
,username=kostya,password=forward,iocharset=utf8,file_mode=0777,dir_mode=0777
```

Для автоматического монтирования диска при загрузке открываем файл /etc/fstab:

```
sudo nano /etc/fstab
```

И записываем в конец файла следующую строку:

```
//192.168.0.8/BOOKS /mnt/kostyabook cifs _netdev
,username=kostya,password=forward,iocharset=utf8,file_mode=0777,dir_mode=0777
       0 0
```

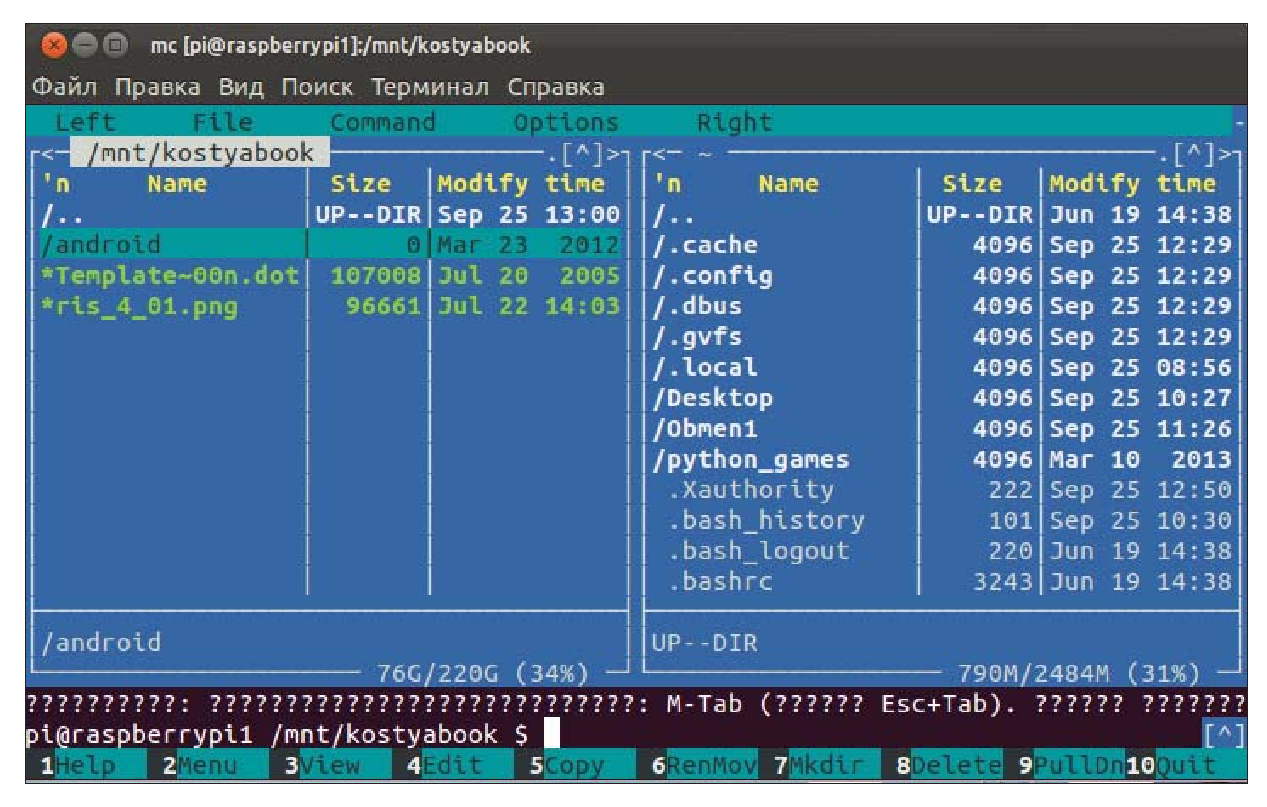


Рис. 4.11. Автоматическое монтирование сетевой папки Windows

После перезагрузки сетевая папка Windows монтируется в каталог /mnt/kostyabook (рис. 4.11).

4.7. Подключение Яндекс.Диска

Продолжим тему расширения дискового пространства. В *главе 3* мы уже рассмотрели подключение к Raspbian флеш-накопителя USB и внешнего жесткого диска USB. Если места не хватает, можно воспользоваться внешними файлохранилищами. Очень удобное средство — Яндекс.Диск. Яндекс.Диск — это так называемый *облачный* сервис, который позволяет вам бесплатно хранить файлы на серверах Яндекса. Вы можете работать с файлами этого Диска на любом устройстве, подключенном к Интернету. Яндекс.Диск поддерживает и протокол WebDAV, что позволяет вам работать с Диском через любой WebDAV-клиент и подключать Диск как сетевой диск Windows. Кроме того, Яндекс.Диск можно использовать для быстрого обмена файлами с другими пользователями Сети — достаточно отправить ссылку на этот файл, и другие пользователи смогут его скачать.

Готовить совместные мероприятия удобно, когда у всех участников есть доступ к одним и тем же файлам. Создавайте общие папки на Яндекс.Диске и приглашайте друзей, сокурсников и коллег для совместной работы над проектами, обмена учебными материалами, а также для планирования путешествий или праздников. Вы редактируете файл у себя на компьютере — изменения через Яндекс.Диск автоматически отражаются у всех. Храните файлы на Диске, и вы сможете просматривать и редактировать их на любом устройстве и в любой точке мира, где есть Интернет: дома, на работе, в кафе и даже в пути. Начните писать отчет в офисе, сохраните на

Диск, откройте на домашнем компьютере — и продолжите с того же места, где остановились.

Для того чтобы подключиться к Яндекс.Диску через WebDAV-клиент, укажите при настройке программы следующие параметры:

- адрес сервера: **https://webdav.yandex.ru**;
- логин (имя пользователя): ваш логин на Яндексе;
- пароль: ваш пароль на Яндексе.

Для того чтобы Raspbian умел обращаться к Яндекс.Диску по протоколу WebDAV, нужно установить пакет `davfs2`:

```
sudo apt-get install davfs2
```

Затем нужно создать папку, куда будет отображаться содержимое Яндекс.Диска:

```
sudo mkdir /mnt/yandex.disk
sudo chmod -R 777 /mnt/yandex.disk
```

Монтируем:

```
sudo mount -t davfs https://webdav.yandex.ru /mnt/yandex.disk -o uid=pi,
gid=pi,rw,file_mode=0777,dir_mode=0777
```

Требуется ввести логин от своего аккаунта в Яндекс и пароль (рис. 4.12).

```
pi@raspberrypi1: ~
pi@raspberrypi1 ~ $ sudo mount -t davfs https://webdav.yandex.ru /mnt/yandex.disk -o uid=pi,g
id=pi,rw,file_mode=0777,dir_mode=0777
Please enter the username to authenticate with server
https://webdav.yandex.ru or hit enter for none.
  Username: victoruni
Please enter the password to authenticate user victoruni with server
https://webdav.yandex.ru or hit enter for none.
  Password:
/sbin/mount.davfs: https://webdav.yandex.ru is already mounted on /mnt/yandex.disk
pi@raspberrypi1 ~ $
```

Рис. 4.12. Монтирование Яндекс.Диска

Теперь настроим автоматическое монтирование Яндекс.Диска при загрузке. Перенастроим для этого пакет `davfs2`:

```
sudo dpkg-reconfigure davfs2
```

В открывшемся интерфейсе (рис. 4.13) выбираем **Yes**. Этим мы позволим монтировать систему непривилегированным пользователям ОС.

Добавим нашего пользователя в группу `davfs2`:

```
sudo usermod -aG davfs2 pi
```

Здесь `pi` — имя пользователя. Если вы меняли стандартное имя, то впишите вместо `pi` имя вашего пользователя.

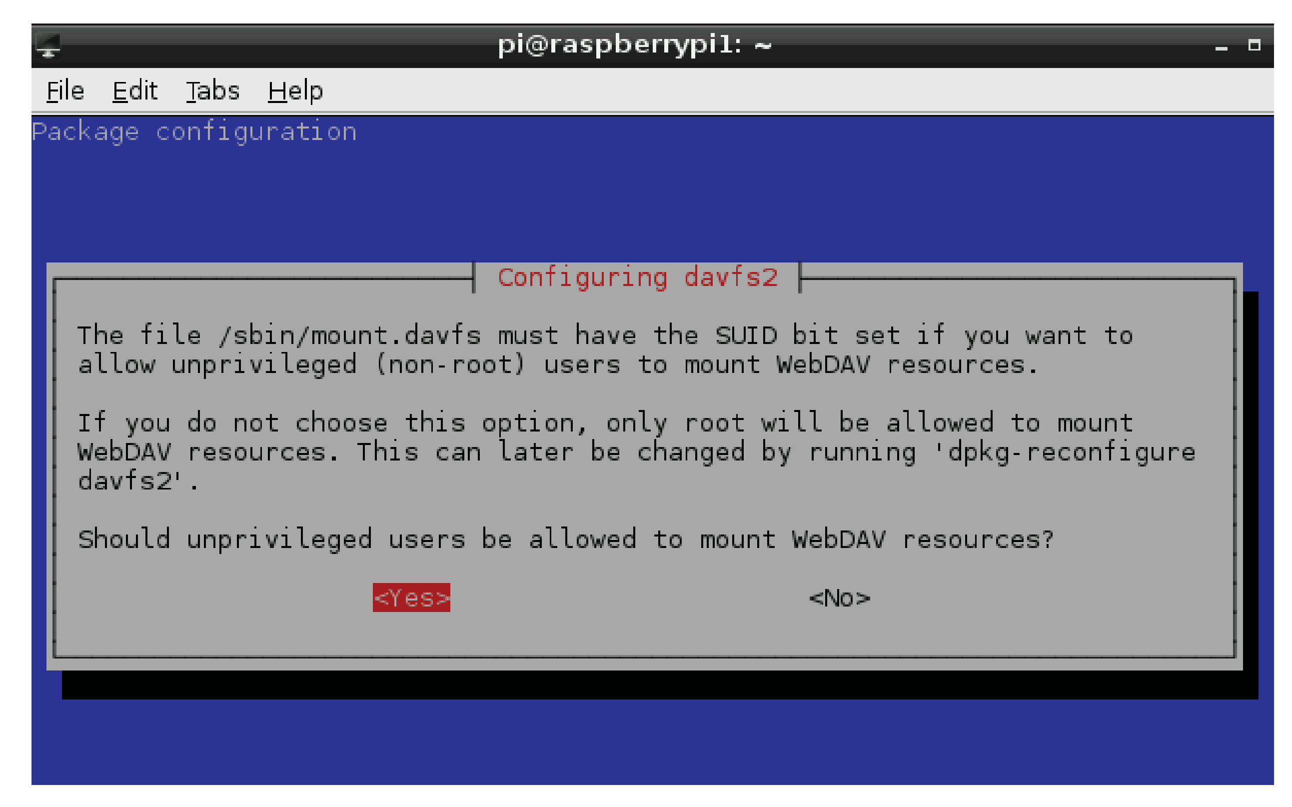


Рис. 4.13. Перенастройка пакета `davfs2`

Настроим параметры авторизации:

```
sudo nano /etc/davfs2/secrets
```

Откроется файл, в конец которого надо добавить запись формата:

```
https://dav-pocket.appspot.com/docso [Логин, выданный сайтом]
                                     [пароль, выданный сайтом]
```

Примечание

Логин и пароль нужно вводить без квадратных скобок.

То есть, добавляем строку:

```
https://webdav.yandex.ru login password
```

где:

- `login` — логин аккаунта Яндекс;
- `password` — пароль.

Сохраняем файл: <Ctrl>+<O> и закрываем его: <Ctrl>+<X>.

Настраиваем автоматическое монтирование. Для этого следует отредактировать файл fstab. Откроем его:

```
sudo nano /etc/fstab
```

и допишем следующую строку:

```
https://webdav.yandex.ru /mnt/yandex.disk davfs uid=1000,user,rw,_netdev 0 0
```

Теперь осталось только перезагрузить Raspbian для проверки автоматического монтирования:

```
sudo reboot
```

Если все сделано правильно, то в папке, которую вы создали и выбрали как точку монтирования, должны появиться файлы, находящиеся на Яндекс.Диске (рис. 4.14). Эти файлы не занимают места на SD-карте, но при этом остаются доступными для редактирования.

Рис. 4.14. Монтирование Яндекс.Диска

4.8. FTP-сервер

FTP, второй по популярности в Интернете протокол после HTTP, обеспечивает обмен файлами между компьютерами. Да, он предназначен только для передачи файлов, зато делает это хорошо. Ставим FTP-сервер:

```
sudo apt-get install proftpd
```

Теперь идем в каталог /etc/proftpd и правим файл proftpd.conf:

```
sudo nano /etc/proftpd/proftpd.conf
```

Изменяем параметр `ServerType` на `standalone`:

```
ServerType standalone
```

Раскомментируем строку:

```
# DefaultRoot          ~
```

Если нужно, изменяем путь на необходимый нам.

Перезапускаем FTP-сервер:

```
sudo /etc/init.d/proftpd restart
```

Все — можно зайти на наш Raspberry Pi с другого компьютера по FTP (рис. 4.15).

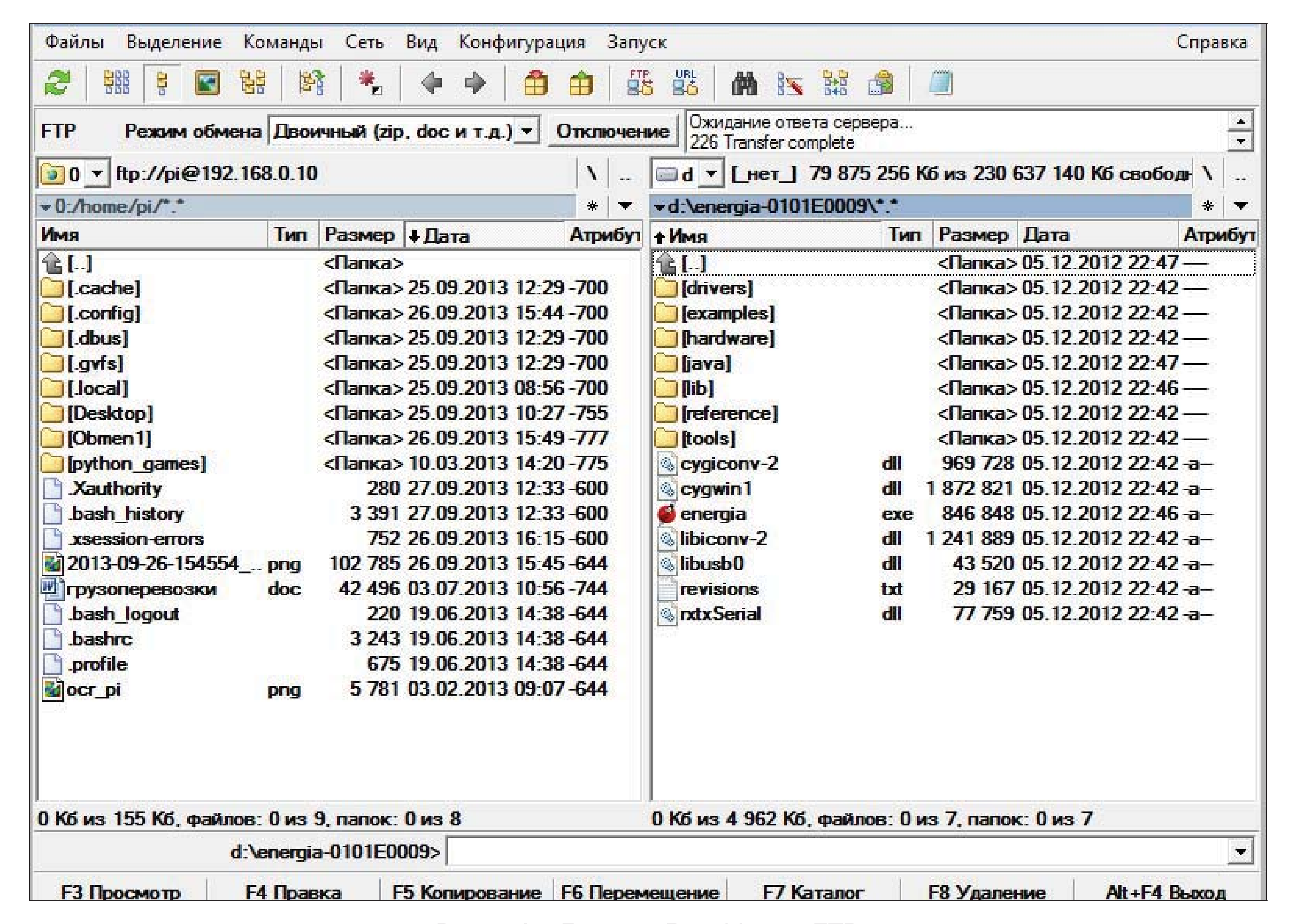


Рис. 4.15. Доступ к Raspbian по FTP

4.9. Веб-сервер

Рассмотрим процесс установки на Raspberry Pi веб-сервера с PHP и MySQL.

Сначала обновим список репозиториев и поиск индексов обновленных версий программ, драйверов, ядра и всего прочего:

```
sudo apt-get update
```

Для доступа по HTTP воспользуемся HTTP-сервером nginx, более быстродейственным и менее "тяжелым", чем Apache. Для установки nginx выполним команду:

```
sudo apt-get -V install nginx
```

Установив nginx, запустим его:

```
sudo /etc/init.d/nginx start
```

Теперь наберем в браузере IP-адрес нашего Raspberry Pi и, если все сделано правильно, увидим стартовую страницу веб-сервера nginx (рис. 4.16).

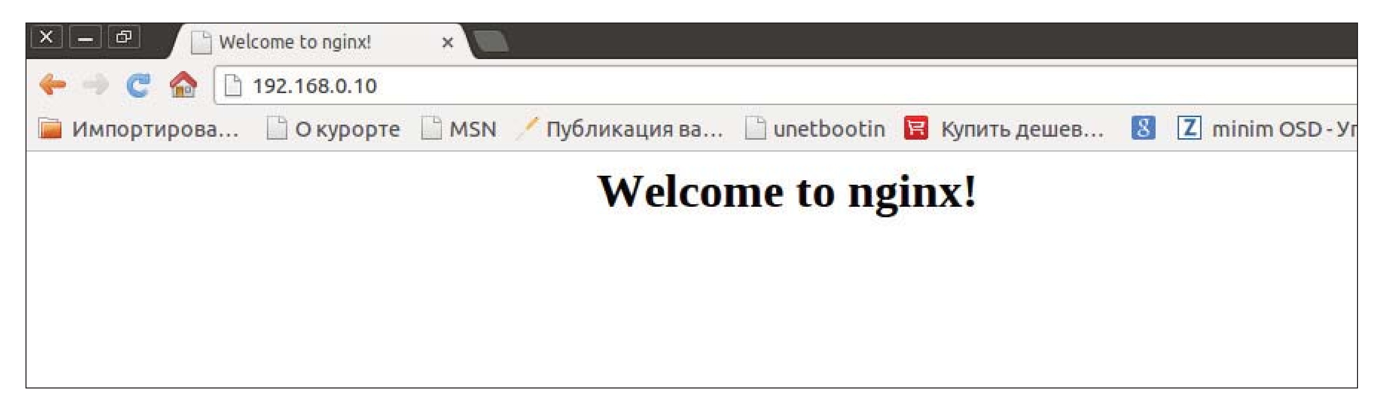


Рис. 4.16. Стартовая страница веб-сервера nginx

Устанавливаем PHP:

```
sudo apt-get install php5-fpm
```

На Raspberry Pi идем в каталог /usr/share/nginx/www и создаем там файл phpinfo.php следующего содержания:

```
<?php
  phpinfo();
?>
```

В файле /etc/nginx/sites-available/default прописываем индексный файл — за это отвечает директива index:

```
index index.php;
```

И раскомментируем строки, касающиеся php5-cgi, чтобы они приобрели следующий вид:

```
location ~ \.php$ {
fastcgi_split_path_info ^(.+\.php)(/.+)$;
# NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
# With php5-cgi alone:
# fastcgi_pass 127.0.0.1:9000;
# With php5-fpm:
fastcgi_pass unix:/var/run/php5-fpm.sock;
fastcgi_index index.php;
include fastcgi_params;
}
```

После этого перезапускаем сервис nginx командой:

```
sudo service nginx restart
```

Перезагружаем nginx и заходим по адресу **http://<ip_raspberry_pi>/test.php**. Если видим страницу, изображенную на рис. 4.17, значит PHP установлен.

Для установки MySQL выполняем команду:

```
sudo apt-get install mysql-server php5-mysql
```

Через некоторое время появится экран (рис. 4.18) с просьбой установить пароль root для пользователя MySQL.

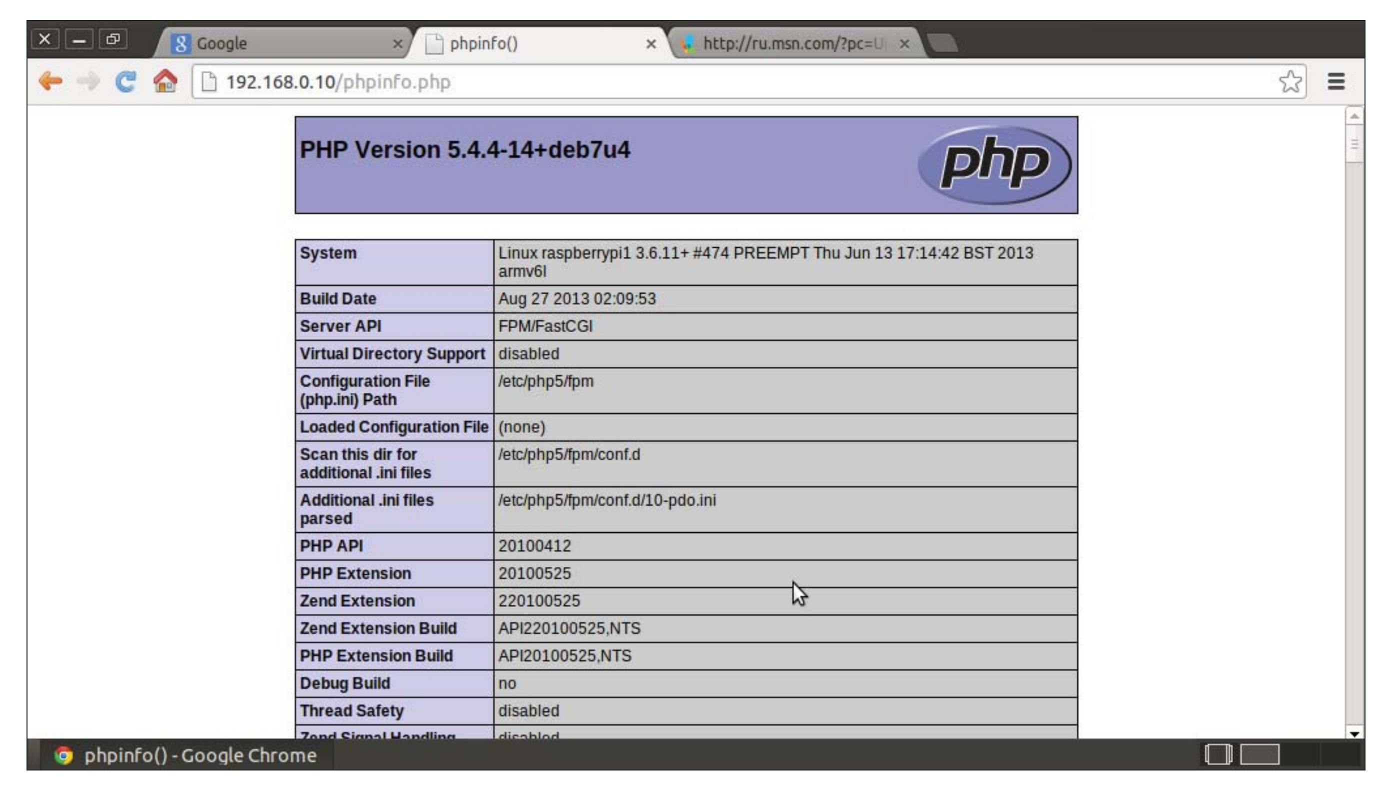


Рис. 4.17. Страница с выводом **phpinfo()**

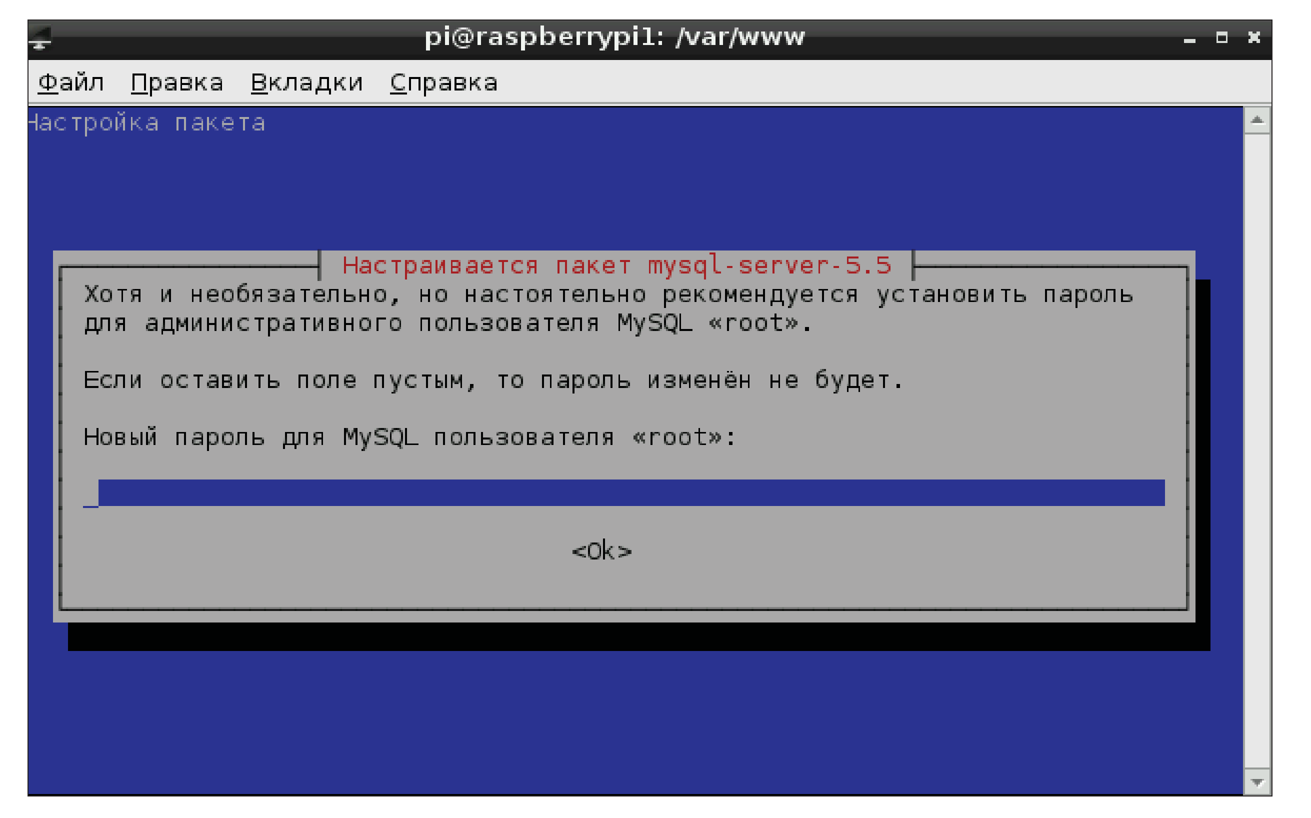


Рис. 4.18. Установка пароля для root пользователя MySQL

После установки MySQL выполняем команду:

```
mysql
```

и попадаем в консольную утилиту для управления базой данных.

Управлять сервером баз данных из командной строки неудобно — предпочтительно работать через веб-интерфейс. Воспользуемся SQL Buddy — веб-интерфейсом для управления MySQL (**http://sourceforge.net/projects/sqlbuddy/**). Скачиваем, разархивируем и копируем в папку, где у вас находятся WWW-файлы. Затем в браузере просто указываем имя этой папки и вводим имя и пароль к MySQL (рис. 4.19).

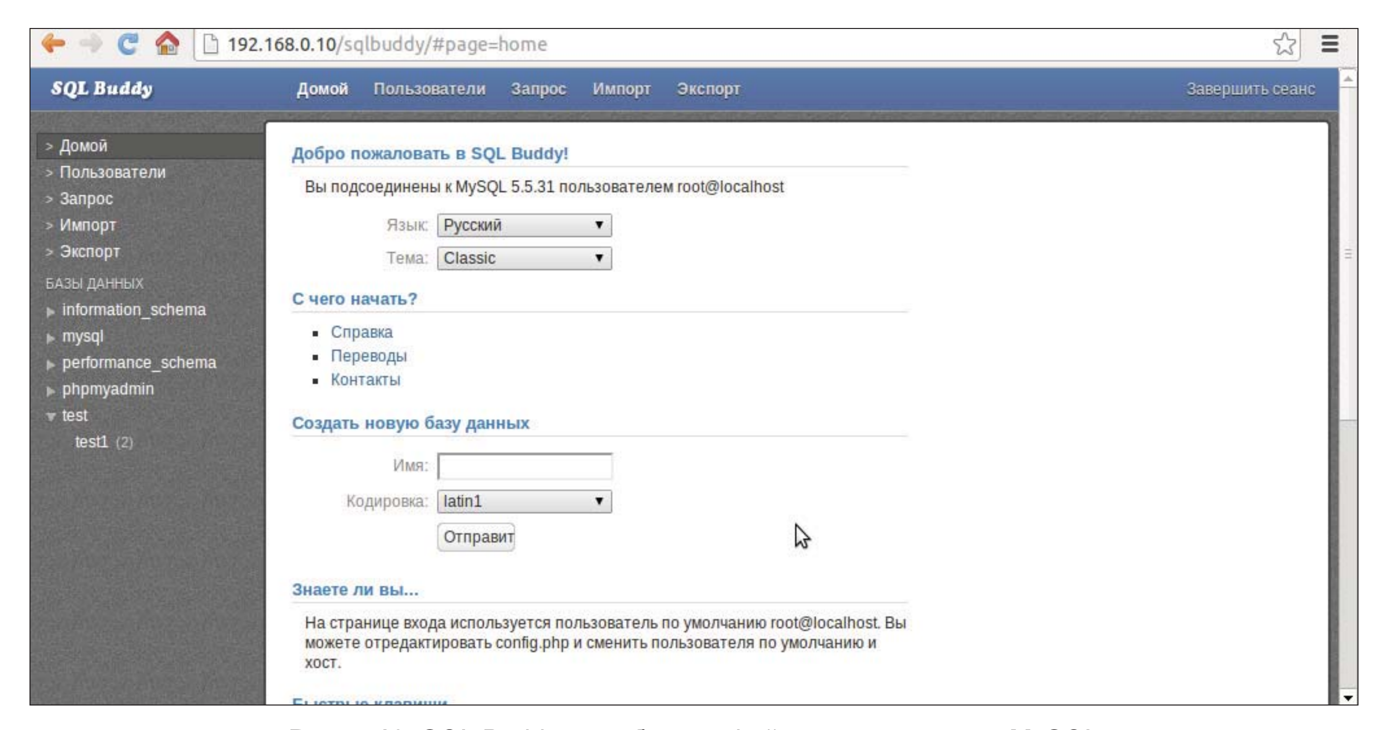


Рис. 4.19. SQL Buddy — веб-интерфейс для управления MySQL

4.10. Торрент-клиент

В качестве торрент-клиента воспользуемся простым и свободным BitTorrent-клиентом Transmission. Transmission, в отличие от многих других BitTorrent-клиентов, отбирает небольшое количество системных ресурсов. В дополнение к оконным интерфейсам Transmission имеет возможность управления через командную строку и веб-браузер.

Для установки Transmission выполним команду:

```
sudo apt-get install transmission-daemon
```

Создадим каталоги для торрентов, закачек, неоконченных закачек в папке torrents на примонтированном внешнем USB-диске (см. *разд. 3.7*) и дадим права на запись:

```
cd /mnt/Arhiv/torrents
sudo mkdir /mnt/flash/torrent
sudo mkdir /mnt/flash/download
sudo mkdir /mnt/flash/incomplete
```

```
sudo chmod 777 /mnt/flash/torrent
sudo chmod 777 /mnt/flash/ download
sudo chmod 777 /mnt/flash/incomplete
```

Теперь настроим Transmission. Для изменения настроек его нужно остановить:

```
sudo /etc/init.d/transmission-daemon stop
```

Затем открыть файл настроек:

```
sudo nano /etc/transmission-daemon/settings.json
```

Основные параметры файла настроек Transmission:

- `download-dir` — каталог закачки: /mnt/Arhiv/torrents/download;
- `incomplete-dir` — каталог неоконченных закачек: /mnt/Arhiv/torrents/incomplete;
- `download-limit` — лимит скорости закачки в Кбит/с;
- `rpc-enabled` — удаленное управление: `true`, если хотим управлять через браузер. Соответственно, устанавливаем значение: `true`;
- `rpc-authentication-required` — аутентификация для удаленного управления;
- `rpc-username` — логин для удаленного управления: `pi`;
- `rpc-password` — пароль для удаленного управления (при запуске торрента пароль шифруется);
- `rpc-whitelist-enabled` — включить список разрешенных IP-адресов: `false`;
- `rpc-port` — порт для доступа к удаленному управлению: `9091`;
- `peer-limit-global` — максимальное количество подключенных пиров на все торренты;
- `peer-limit-per-torrent` — количество пиров на один торрент;
- `download-queue-size` — количество одновременных закачек.

Чтобы управлять Transmission через веб-интерфейс (рис. 4.20), наберите в адресной строке:

```
http://192.168.0.101:9091
```

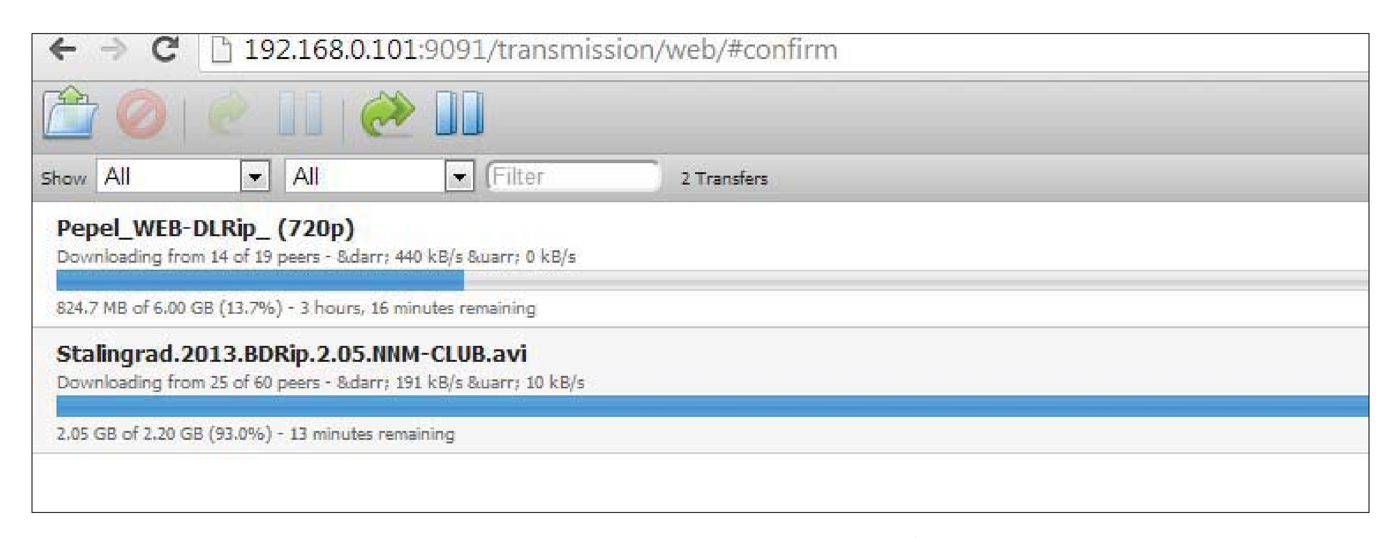


Рис. 4.20. Управление Transmission через веб-интерфейс

Для удаленного управления демоном Transmission по протоколу RPC (вызов удаленных процедур) рекомендую воспользоваться программой Transmission Remote GUI (рис. 4.21). Скачать программу можно по ссылке: **https://code.google.com/p/transmisson-remote-gui/downloads/list**.

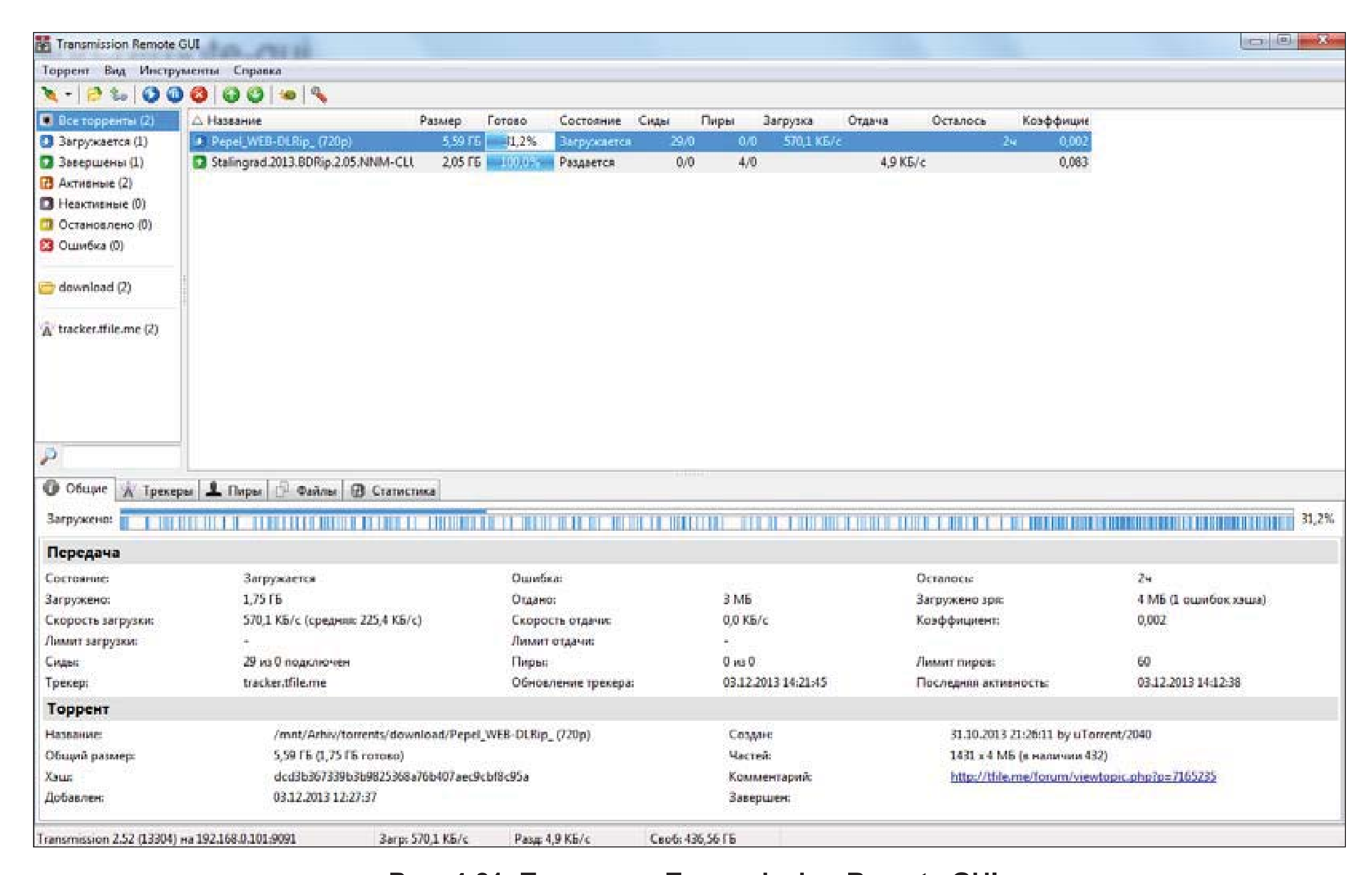


Рис. 4.21. Программа **Transmission Remote GUI**

4.11. Видеотрансляция с помощью веб-камеры

Рассмотрим, как можно использовать Raspberry Pi для трансляции изображения с веб-камеры USB в сеть. Прежде всего подключим к Raspberry Pi веб-камеру USB. У меня в наличии имелись две веб-камеры: Defender Glory 327 и Logitech C270 — подключим сначала Defender Glory 327 (рис. 4.22).

Прежде всего в терминале устанавливаем пакет `libv4l-0`:

```
sudo apt-get install libv4l-0
```

Далее необходимо установить пакет `mjpg-streamer-rpi`:

```
wget http://www.bobtech.ro/get?download=36:mjpg-streamer-rpi
mv get\?download\=36\:mjpg-streamer-rpi mjpg-streamer-rpi.tar.gz
tar -zxvf mjpg-streamer-rpi.tar.gz
```

Параметры потока можно настроить в файле mjpg-streamer.sh:

```
cd mjpg-streamer
sudo nano mjpg-streamer.sh
```



Рис. 4.22. USB веб-камера Defender Glory 327

По умолчанию эти параметры имеют следующие значения:

- `VIDEO_DEV="/dev/video0"` — идентификатор устройства;
- `FRAME_RATE="30"` — частота кадров (FPS);
- `RESOLUTION="640x480"` — разрешение;
- `PORT="8080"` — HTTP-порт;
- `YUV="false"` — флаг YUV-кодирования.

Изменяем необходимые для камеры Defender Glory 327 настройки:

- `FRAME_RATE="1"` — частота кадров (FPS);
- `RESOLUTION="320x240"` — разрешение;
- `YUV="true"` — флаг YUV-кодирования.

Для запуска трансляции запускаем bash-скрипт mjpg-streamer.sh:

```
sudo ./mjpg-streamer.sh start
```

Остановка трансляции осуществляется командой:

```
sudo ./mjpg-streamer.sh stop
```

Видеопоток станет доступен по адресу (URL) следующего вида:

```
http://raspberrypi:8080?action=stream
```

Пример потокового изображения с камеры показан на рис. 4.23.

Надо отметить, что задержки показа изображения по сети достигали у меня 2–3 секунд. Аппаратное ускорение кодирования видео на Raspberry Pi отсутствует, а без него, программно, сжатие идет со скоростью 2–5 fps и на все 100 % загружает машину.



Рис. 4.23. Просмотр потока с веб-камеры через браузер

Пришлось пойти другим путем и применить GStreamer — мощный фреймворк[1] для построения мультимедийных приложений, который позволяет создавать приложения различных уровней сложности, начиная от простого консольного плеера и заканчивая полноценными аудио/видеоплеерами, мультимедийными редакторами и т. п. Мультимедийный программный комплекс GStreamer обладает интерфейсом в виде командной строки.

Для установки GStreamer в терминале выполним команду:

```
sudo apt-get install gstreamer0.10-plugins-base gstreamer0.10-plugins-good
gstreamer0.10-plugins-ugly gstreamer0.10-tools
```

Для запуска потока на конкретный компьютер набираем в терминале:

```
sudo gst-launch-0.10 v4l2src device=/dev/video1 ! 'video/x-raw-
yuv,width=640,height=480, framerate=20/1'! ffmpegcolorspace ! rtpvrawpay !
udpsink host=192.168.0.3 port=5100 sync=false
```

Здесь:

- `device=/dev/video1` — камера-источник картинки;
- `video/x-raw-yuv,width=640,height=480, framerate=20/1` — формат картинки;
- `host=192.168.0.50 port=5100` — IP-адрес компьютера, куда посылается поток.

---

[1] Фреймворк — программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Видео передается по протоколу RTP. Этот протокол работает на транспортном уровне и используется при передаче трафика реального времени. На стороне компьютера-получателя потока в терминале вводим:

```
gst-launch udpsrc caps="application/x-rtp, media=(string)video,
clock-rate=(int)90000, encoding-name=(string)RAW, sampling=(string)YCbCr-4:2:0,
depth=(string)8, width=(string)640, height=(string)480, ssrc=(uint)1825678493,
payload=(int)96, clock-base=(uint)4068866987, seqnum-base=(uint)24582"
port=5100 ! queue ! rtpvrawdepay ! queue ! ffmpegcolorspace ! autovideosink
```

и на компьютере-приемнике должны увидеть поток. Однако передать поток с веб-камеры Defender Glory 327 у меня не получилось, зато камера Logitech C270 (рис. 4.24) с задачей прекрасно справилась. Задержки сигнала при этом практически на было заметно.



Рис. 4.24. USB веб-камера Logitech C270

4.11.1. Сервер видеонаблюдения

Motion — весьма мощная программа для работы с видеокамерой. Кроме всего прочего, она позволяет определять движение в кадре и запускать по этому событию определенные действия. Motion также устанавливает свой веб-сервер, позволяющий удаленно контролировать настройки программы и смотреть видео.

Для установки Motion выполним команду:

```
sudo apt-get install motion
```

Перед конфигурированием веб-сервера нужно прежде всего определить порт, на котором этот сервер заработает. Для этого откроем файл конфигурации и отредактируем его:

```
sudo nano /etc/motion/motion.conf
```

Назначим порты, на которые будет выводиться поток:

```
webcontrol_port 8088
webcam_port 8089
```

Установим разрешение на удаленное администрирование и удаленный доступ:

```
webcam_localhost off
control_localhost off
```

И зададим логин-пароль для доступа:

```
control_authentication login:password
```

Большинство современных веб-камер поддерживают передачу сжатого видео. Для того чтобы включить этот режим, необходимо найти в файле конфигурации строчку `V412_palette 8` и заменить `8` на `2`. Если камера не поддерживает этот режим, Motion при запуске предупредит вас о несоответствии форматов и переключится в поддерживаемый формат.

Плюсом Motion является встроенный датчик движения. То есть нам не обязательно вести запись с камеры постоянно — пока движения в зоне наблюдения нет, запись не ведется, но как только там появится движение, Motion начнет запись:

```
output_all off
```

Можно запустить Motion как сервис, для этого установим параметр:

```
daemon on
```

Для запуска Motion набираем в терминале:

```
sudo motion
```

Сохранение снимков производится в каталог /tmp/motion (рис. 4.25). Учитывая небольшой объем дискового пространства Raspberry Pi, логично переопределить каталог сохранения:

```
target_dir /tmp/motion
```

Видеопоток Motion (рис. 4.26) доступен по адресу:

```
http://Raspberry_IP: stream_port
```

Управление Motion (рис. 4.27) доступно по адресу:

```
http://Raspberry_IP: webcontrol_port
```

Одной из интересных особенностей Motion является возможность установить обработчики событий: обнаружено движение — `on_event_start`, и картинка сохранена — `on_picture_save`. При обнаружении движения также можно отправлять уведомление на e-mail.

Чтобы реализовать эту возможность, установим утилиту `ssmtp`, которая позволяет слать почту через relay-сервер или через обычный SMTP-аккаунт на стороннем сервере:

```
sudo apt-get install ssmtp
```

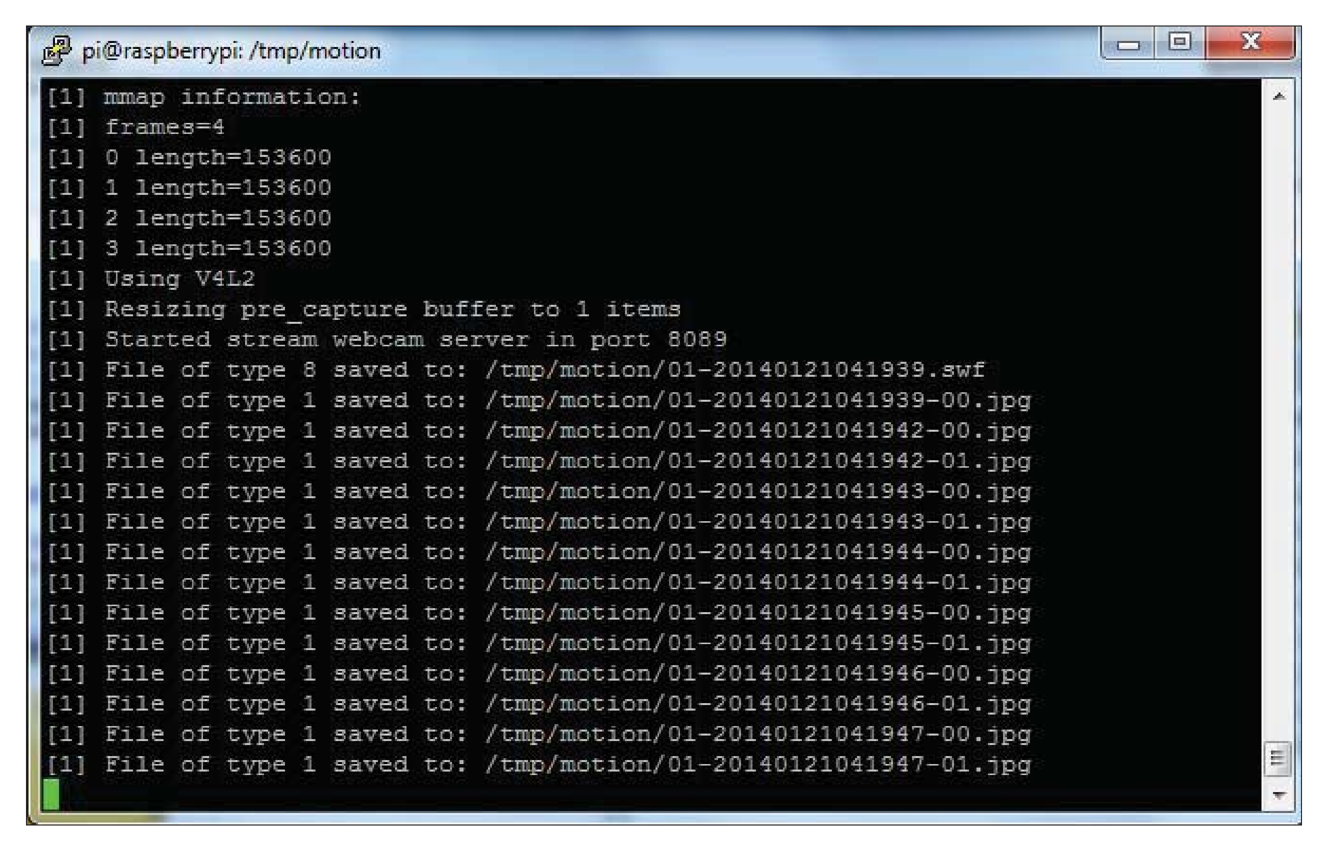


Рис. 4.25. Сохранение снимков Motion

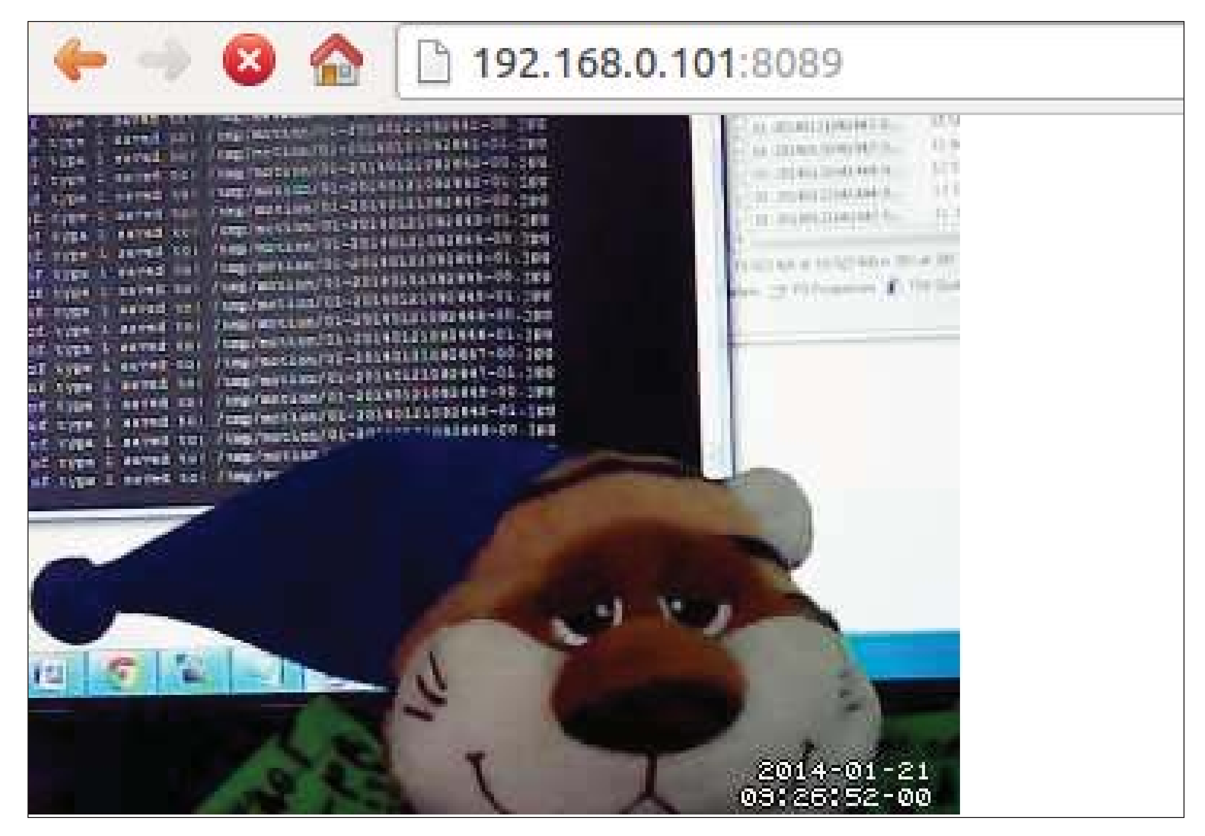


Рис. 4.26. Просмотр видеопотока Motion



Рис. 4.27. Страница управления Motion

Для редактирования параметров утилиты откроем ее конфигурационный файл:

```
sudo nano /etc/ssmtp/ssmtp.conf
```

и установим следующие параметры (листинг 4.1):

Листинг 4.1. Параметры конфигурационного файла утилиты ssmtp

```
# Указываем имя пользователя на smtp-сервере
root=victor.petin@gmail.com

# Указываем smtp-сервер, через который будем отправлять письма
mailhub=smtp.gmail.com:465

# Выставляем принудительное переписывание домена в поле From:
# В противном случае наше письмо не будет отправлено сторонним
# smtp-сервером
rewriteDomain=gmail.com

# запрещаем скриптам "решать", с какого ящика они отправляют письмо.
# Поле From: будет выставляться самим smtp.
FromLineOverride=NO

UseTLS=YES
# логин - пароль
AuthUser=victor.petin@gmail.com
AuthPass=*********
```

Создадим скрипт send_motion.sh:

```
sudo touch /usr/bin/send_motion.sh
```

и запишем в него следующий код:

```
#!/bin/sh
echo "motion -$(date)!!!" | ssmtp online-spravka@bk.ru
```

Сделаем файл исполняемым:

```
chmod +x /usr/bin/send_motion.sh
```

Затем нужно вписать в файл настроек Motion следующую строку:

```
on_event_start /usr/bin/send_motion.sh
```

Теперь при движении в зоне наблюдения камеры к вам на почту придет уведомление.

4.11.2. Передача потокового видео с камеры Raspberry Pi Camera Board

Рассмотрим, как передавать потоковое видео с камеры Raspberry Pi Camera Board (см. *разд. 3.11*). Поскольку эта камера подключена напрямую к графическому процессору через CSi-разъем на плате, запись и кодирование видео происходят без использования процессорного времени.

Установим пакет MJPG-streamer:

```
su pi
https://svn.code.sf.net/p/mjpg-streamer/code/mjpg-streamer/ mjpg-streamer
cd mjpg-streamer/mjpg-streamer
make USE_LIBV4L2=true clean all
sudo su
make DESTDIR=/usr install
cp -R www /var/
```

Напишем sh-скрипт для запуска камеры. Для этого создадим файл camera.sh:

```
sudo nano /usr/local/bin/webcamera.sh
sudo chmod +x /usr/local/bin/webcamera.sh
```

Содержимое скрипта представлено в листинге 4.2.

Листинг 4.2. Скрипт для запуска камеры

```
#!/bin/bash
if [ -d /tmp/stream ];then
        echo "/tmp/stream already created"
else
        mkdir /tmp/stream
fi

if [ -f /tmp/stream/pic.jpg ];then
        echo "raspistill already running"
else
        raspistill -w 640 -h 480 -q 5 -o /tmp/stream/pic.jpg -tl 100 -t
9999999&
fi
mjpg_streamer -i "input_file.so -f /tmp/stream" -o "output_http.so -w /var/www"
```

Запуск скрипта осуществляется командой:

```
/usr/local/bin/camera.sh
```

Для просмотра изображения с камеры можно подключиться к веб-странице, которую создает MJPG-streamer (рис. 4.28).



Рис. 4.28. Страница просмотра потокового видео с камеры Raspberry Pi Camera Board

4.12. Синтез речи на Raspberry Pi

Синтезирование голоса производится с помощью специальных программ-синтезаторов, самые известные из которых Festival и Espeak. Наиболее качественным из синтезаторов считается Festival, разрабатываемый сотрудниками многих университетов по всему миру и оснащенный поддержкой русского языка (правда, не "из коробки"). У синтезатора Espeak более низкое качество произношения, но он компактнее, чем Festival, и работает быстрее, что немаловажно для нашего мини-компьютера.

4.12.1. Голосовой синтезатор Espeak

Установим программу Espeak:

```
apt-get install espeak
```

Для запуска воспроизведения речи выполним команду:

```
espeak -ven -a100 -s200 "Hello"
```

Основные параметры команды `espeak`:

- ❒ `-a` — амплитуда воспроизведения (0–200);
- ❒ `-s` — скорость воспроизведения (80–450);
- ❒ `-v` — язык произношения (`ru` — русский);
- ❒ `-f` — текстовый файл для преобразования в речь;
- ❒ `-w` — имя звукового файла (WAV) для воспроизведения.

Команда произнесения фразы на русском языке:

```
espeak -vru -a200 -s200 "Привет всем"
```

Команда преобразования текста из файла text.txt в речь:

```
espeak -vru -a200 -s200 -f text1.txt
```

При этом текст воспроизводится, но выдается ошибка (рис. 4.29).

```
pi@raspberrypi2 ~ $ espeak -vru -a200 -s100 -p10 -f text1.txt
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.front
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.rear
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.center_lfe
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.side
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround40
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround41
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround50
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround51
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.surround71
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.iec958
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib pcm.c:2217:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib pcm_dmix.c:957:(snd_pcm_dmix_open) The dmix plugin supports only playba
ck stream
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jack server is not running or cannot be started
pi@raspberrypi2 ~ $
```

Рис. 4.29. Ошибка при воспроизведении программой Espeak

Ошибка убирается при перенаправлении вывода: `2>/dev/null`:

```
espeak -vru -a200 -s200 -f text1.txt 2>/dev/null
```

4.12.2. Голосовое оповещение о приходящих письмах на почту *gmail.com*

Создадим небольшой проект голосового оповещения при приходе новых писем на почту **gmail.com**. Для проверки почты (один раз в 10 минут) и голосового оповеще-

ния о новых письмах, поступивших для учетной записи на **gmail.com**, напишем на языке Python скрипт, использующий интерфейс Gmail APIs[1] и учитывающий, что информация о непрочитанных сообщениях выдается в формате XML при обращении к странице: **https://LOGIN:@PASSWORD@mail.google.com/mail/feed/atom** (рис. 4.30).

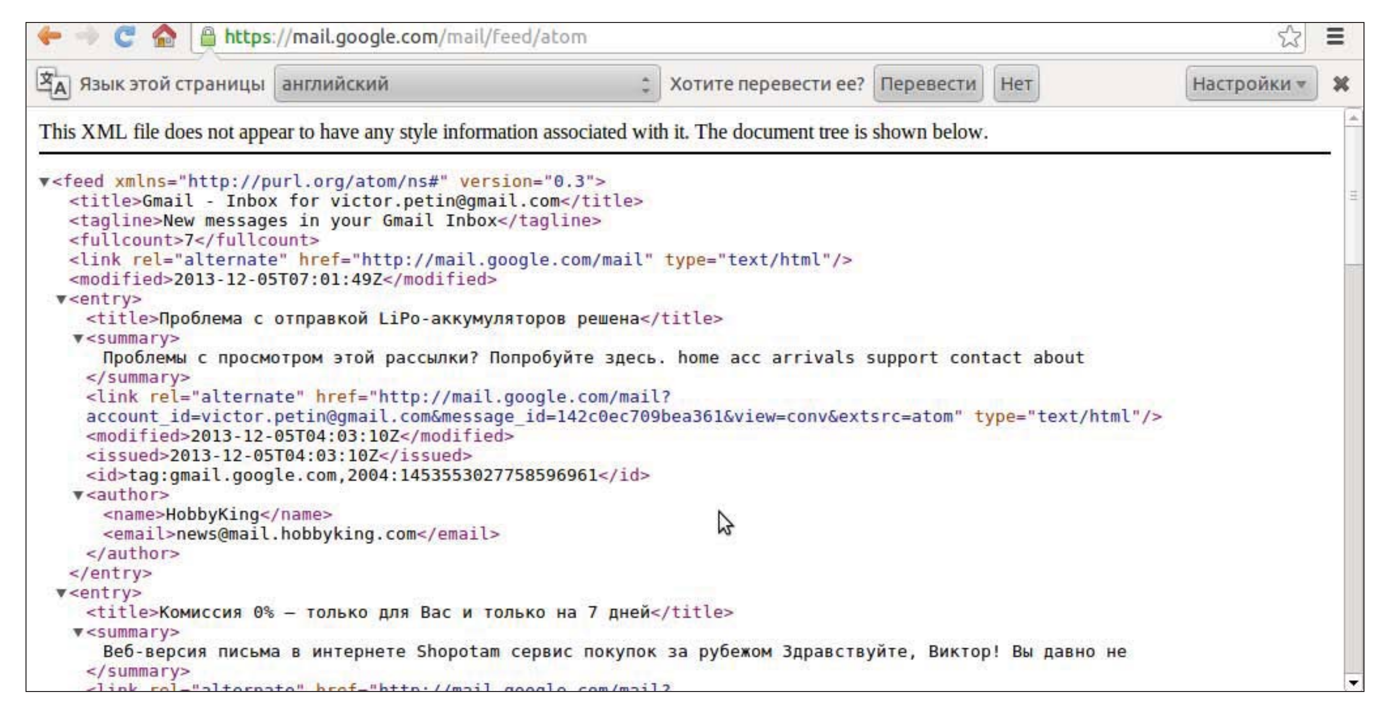


Рис. 4.30. Информация о непрочитанных сообщениях

Для написания скрипта на Python мы воспользуемся модулем `feedparser`. Самый простой способ устанавливать модули Python — с помощью менеджера пакетов `python-pip`. Для его установки выполним команду:

```
sudo apt-get install python-pip python2.7-dev
```

Чтобы `python-pip` правильно заработал, нужно обновить пакет `distribute` программой `easy_install`:

```
sudo easy_install -U distribute
```

Теперь с помощью менеджера пакетов `python-pip` установим модуль `feedparser`:

```
sudo pip install feedparser
```

Скрипт получает XML-файл и проверяет, появились ли новые письма. Если появились, то формирует строку для голосового сообщения и произносит ее с помощью синтезатора Espeak (см. *разд. 4.12.1*). Кроме того, скрипт формирует HTML-файл со списком новых писем и ссылками для перехода на них. Потом этот HTML-файл открывается в браузере (рис. 4.31). Содержимое скрипта (файл gmail_espeak.py) представлено в листинге 4.3.

[1] Подробную информацию о Gmail APIs можно найти по адресу: **https://developers.google.com/gmail/gmail_inbox_feed**.

1. Receipt for your payment to liuxiang8866@hotmail.com
2. Confirmation of your order of U-disk audio player SD card voice module MP3 Sound module WTV020-SD-16P Arduino...
3. You have won eBay Item 310629013078 U-disk audio player SD card voice module MP3 Sound module WTV020-SD-16P Arduino !
4. Проблема с отправкой LiPo-аккумуляторов решена
5. Комиссия 0% — только для Вас и только на 7 дней
6. Adding subscription IDs to ROSBridge-published messages
7. Now In Stock: The NEW Spektrum DX4R PRO
8. BH Group Co. Ltd.: информация по нагрузке lermontov-kmv.ru на сервер
9. Now In Stock: Brushless ECX Trucks

Рис. 4.31. Страница со ссылками на новые письма

Листинг 4.3. Скрипт gmail_espeak.py

```
#!/usr/bin/env python
#-*-coding:utf-8 -*-
#  Скрипт проверки новых писем
#  на почте gmail и оповещение голосом (espeak)
import subprocess
import shlex
import feedparser
import webbrowser

USERNAME="victor.petin@gmail.com"
PASSWORD="password"
DIR="/home/pi/python_prg/gmail_espeak/"

ff=feedparser.parse("https://" + USERNAME + ":" + PASSWORD +
"@mail.google.com/gmail/feed/atom");
count_letters = int(ff["feed"]["fullcount"])
if count_letters > 0:
    print("Есть новые письма")
    command1='espeak -vru -a150 -s80 "Многоуважаемый хозяин, У вас '
    if count_letters>9 and count_letters<20:
       command1+=str(count_letters)
       command1+=' новых  писем"'
    elif count_letters%10==1:
       if(count_letters>10):
          command1+=str(int(count_letters/10)*10)
       command1+=" одно"
       command1+=' новое письмо"'
    elif count_letters%10>4 or count_letters%10==0:
       if(count_letters>10):
          command1+=str(int(count_letters/10)*10)
       command1+=str(count_letters%10)
       command1+=' новых писем"'
    else:
       if(count_letters>10):
          command1+=str(int(count_letters/10)*10)
```

```
        command1+=str(count_letters%10)
        command1+=' новых письма"'
    fhtml=open(DIR+"links.html","w")
    for i in range(0,len(ff["entries"])):
        str2=str(i+1)+". ";
        str2+="<a href='"+ff["entries"][i].link+"'>";
        str2+=ff["entries"][i].title+"</a><br>";
        print str2
        fhtml.write(str2.encode('windows-1251'))

    fhtml.close()
    command1+=' 2>/dev/null'
    webbrowser.open(DIR+"links.html")
    subprocess.Popen(command1,shell=True).communicate()
```

Теперь осталось только добавить в планировщик cron задачу для запуска нашего скрипта каждые 10 минут. Открываем для редактирования файл конфигурации cron:

```
sudo crontab -e
```

И добавляем в конец файла строку:

```
* * * * * python /home/pi/python_prg/gmail_espeak/gmail_espeak.py
```

Теперь каждые 10 минут почта проверяется и, в случае наличия новых сообщений, выдается голосовое оповещение и в браузер выводится страница со ссылками перехода на новые письма.

> ***Примечание***
> Код скрипта *gmail_espeak.py* вы найдете в папке *glava_04\gmail_espeak* сопровождающего книгу электронного архива (см. *приложение*).

4.13. Raspberry Pi и голосовое управление

Создадим программу голосового управления нашим Raspberry Pi (на дистрибутиве Raspbian). Raspberry Pi не имеет специального входа для микрофона, но можно подключить к нему USB-микрофон или веб-камеру USB со встроенным микрофоном, — например, имеющуюся у меня веб-камеру Logitech C270 (см. рис. 4.24).

Устанавливаем необходимые пакеты:

```
sudo apt-get install v4l-utils sox alsa-tools alsa-oss flac
```

Выводим список USB-устройств:

```
lsusb
```

Результат вывода приведен на рис. 4.32. Наша веб-камера определена в нем так: **Bus 001 Device 009: ID 046d:0825 Logitech, Inc. Webcam C270**.

```
pi@raspberrypi2 ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 008: ID 1a40:0201 Terminus Technology Inc. FE 2.1 7-port Hub
Bus 001 Device 005: ID 045e:00db Microsoft Corp. Natural Ergonomic Keyboard 4000 V1.0
Bus 001 Device 009: ID 046d:0825 Logitech, Inc. Webcam C270
Bus 001 Device 010: ID 1058:1023 Western Digital Technologies, Inc. Elements SE
pi@raspberrypi2 ~ $ cat /proc/asound/cards
 0 [ALSA           ]: BRCM bcm2835 ALSbcm2835 ALSA - bcm2835 ALSA
                      bcm2835 ALSA
 1 [U0x46d0x825    ]: USB-Audio - USB Device 0x46d:0x825
                      USB Device 0x46d:0x825 at usb-bcm2708_usb-1.2.6, high speed
pi@raspberrypi2 ~ $
```

Рис. 4.32. Список USB-устройств

Смотрим теперь список аудиоустройств:

```
cat /proc/asound/cards
```

4.13.1. Движок распознавания речи Julius

Для преобразования речи в текст можно использовать сервис Google Speech. Но здесь есть свои минусы:

- требуется подключение к Интернету;
- процент распознавания не очень высок (хотя при использовании Google Speech API с планшета распознавание производится гораздо лучше);
- задержка с получением ответа 3–5 секунд.

Так что лучше мы воспользуемся движком распознавания речи японского происхождения Julius. Для работы с Julius требуются предварительно обученные грамматическая и акустическая модели. Акустическая модель строится путем обработки звуковых файлов (с начитанными человеком фрагментами текстов) специальными программами. Наиболее правильно самому наговорить эти фрагменты и, таким образом, научить движок распознавать именно свой голос (в том числе интонацию и ошибки произношения) и словосочетания, которые нужны именно вам. Тогда процент правильного распознавания будет стремиться к значению 100. Однако все это сложно, требует определенной подготовки и времени, поэтому здесь мы ограничимся акустической моделью, распространяемой с сайта **http://www.voxforge.org**. Приемлемая акустическая модель имеется там только для английского языка, для русского модели пока нет. Но для нашей задачи достаточно распознавания английской речи.

Итак, приступим к установке Julius.

Сначала доустановим необходимые пакеты:

```
sudo apt-get install alsa-tools alsa-oss flex zlib1g-dev libc-bin libc-dev-bin
python-pexpect libasound2 libasound2-dev cvs
```

Сделаем тестовую проверку записи на микрофон и воспроизведения:

```
arecord -d 10 -D plughw:1,0 test.wav
aplay test.wav
```

Установим пакет Julius из SNV:

```
cvs -z3 -d:pserver:anonymous@cvs.sourceforge.jp:/cvsroot/julius co julius4
```

Определим переменные окружения:

```
export CFLAGS="-O2 -mcpu=arm1176jzf-s -mfpu=vfp -mfloat-abi=hard -pipe -fomit-frame-pointer"
```

Далее конфигурирование, компиляция и построение пакета:

```
cd julius4
./configure --with-mictype=alsa
make
sudo make install
```

Здесь надо отметить, что конфигурирование с параметром `--with-mictype=alsa` не проходило, и было сделано без него:

```
./configure
make
sudo make install
```

Этот процесс достаточно долгий...

Теперь скачаем упомянутую ранее английскую акустическую модель:

http://www.repository.voxforge1.org/downloads/Main/Tags/Releases/0_1_1-build726/Julius_AcousticModels_8kHz-16bit_MFCC_O_D_(0_1_1-build726) (1).zip

и распакуем ее в папку julius_model2.

Затем можно приступить к настройке, которая фактически включает в себя только процесс создания словаря: списка слов, который должен уметь распознавать движок, и объяснение того, как эти слова могут между собой сопоставляться. Нужно это для двух целей: во-первых, движок должен знать произношение слов и понимать их, а, во-вторых, сократив словарь всего до нескольких фраз, мы значительно повысим качество распознавания.

Нам необходимо поменять содержимое следующих файлов:

- sample.voca — содержит совсем небольшой список слов, а также их фонетическое представление (что-то вроде транскрипции);
- sample.grammar — содержит правила, в каких комбинациях эти слова могут быть использованы.

Возможностями Julius можно воспользоваться, например, для управления роботом с помощью голосовых команд. Соответственно, составим и словарь.

Сначала определимся со списком слов для голосовых команд управления роботом:

- выбор объекта, к которому обращены команды: "Robert" — сам робот;
- команды приветствия: "Hi", "Buy", "Thanks", "Sleep";

- команды выбора действия: "On", "Off", "Forward", "Back", "Left", "Right", "Turn" (поворот), "Dance" (набор предопределенных движений), "Base" (на базу);
- команды — численные параметры выбора действия: "One"–"Nine" (1–9) — скорость движения (или номер танца);
- команды — относительные параметры выбора действия: "Slow", "Fast" — быстрее, медленнее (для скорости).

Содержимое файла sample.voca представлено в листинге 4.4.

Листинг 4.4. Файл sample.voca

```
% NS_B
<s>         sil

% NS_E
</s>         sil

% NAME
ROBERT        r ow b er t

% HELLO

HI            hh ay
BUY           b ay
THANKS        th ae ng k s
SLEEP         s l iy p

% DO

ON           ao n
OFF          ao f
BASE         b ey s

FORWARD      f ao r w aa r d
BACK         b eh k
LEFT         l eh f t
RIGHT        r ay t
TURN         t er n
DANCE        d aa n s

% DIGIT

ONE         w ah n
TWO         t uw
THREE       th r iy
FOUR        f ao r
```

```
FIVE        f ay v
SIX         s ih k s
SEVEN       s eh v ax n
EIGHT       ey t
NINE        n ay n


% SPEED

FAST        f aa s t
SLOW        s l ow
```

Фонетическое представление слов содержится в файле beep-1.0, который находится в папке Glava_04\julius сопровождающего книгу электронного архива (см. *приложение*).

Содержимое файла sample.grammar с правилами, определяющими в каких комбинациях слова команд могут быть использованы, представлено в листинге 4.5.

Листинг 4.5. Файл sample.grammar

```
S : NS_B FRAZA NS_E

FRAZA: NAME HELLO
FRAZA: NAME DO
FRAZA: NAME DO DIGIT
FRAZA: NAME DO SPEED
FRAZA: HELLO
FRAZA: DO
FRAZA: DIGIT
FRAZA: SPEED
```

Выполняем команду для генерации файлов sample.dfa, sample.term и sample.dict:

```
mkdfa sample
```

Теперь можно приступать к проверке работы языковой модели. Настроим ALSEDEV для записи голоса с микрофона:

```
julius -input mic -C julius.jconf
```

Затем произносим фразы и смотрим результат распознавания (рис. 4.33).

Как можно видеть, распознавание осуществляется хуже, чем на нетбуке с системой Ubuntu 11.10, и совсем не распознаются фразы... Впрочем, поупражнявшись с произношением, можно прийти к приемлемому результату.

Напишем скрипт на Python, выделяющий распознаваемое слово из скрипта распознавания. Нас интересует слово в строке, начинающейся со слова **sentence1** между тегами <s> и <s>.

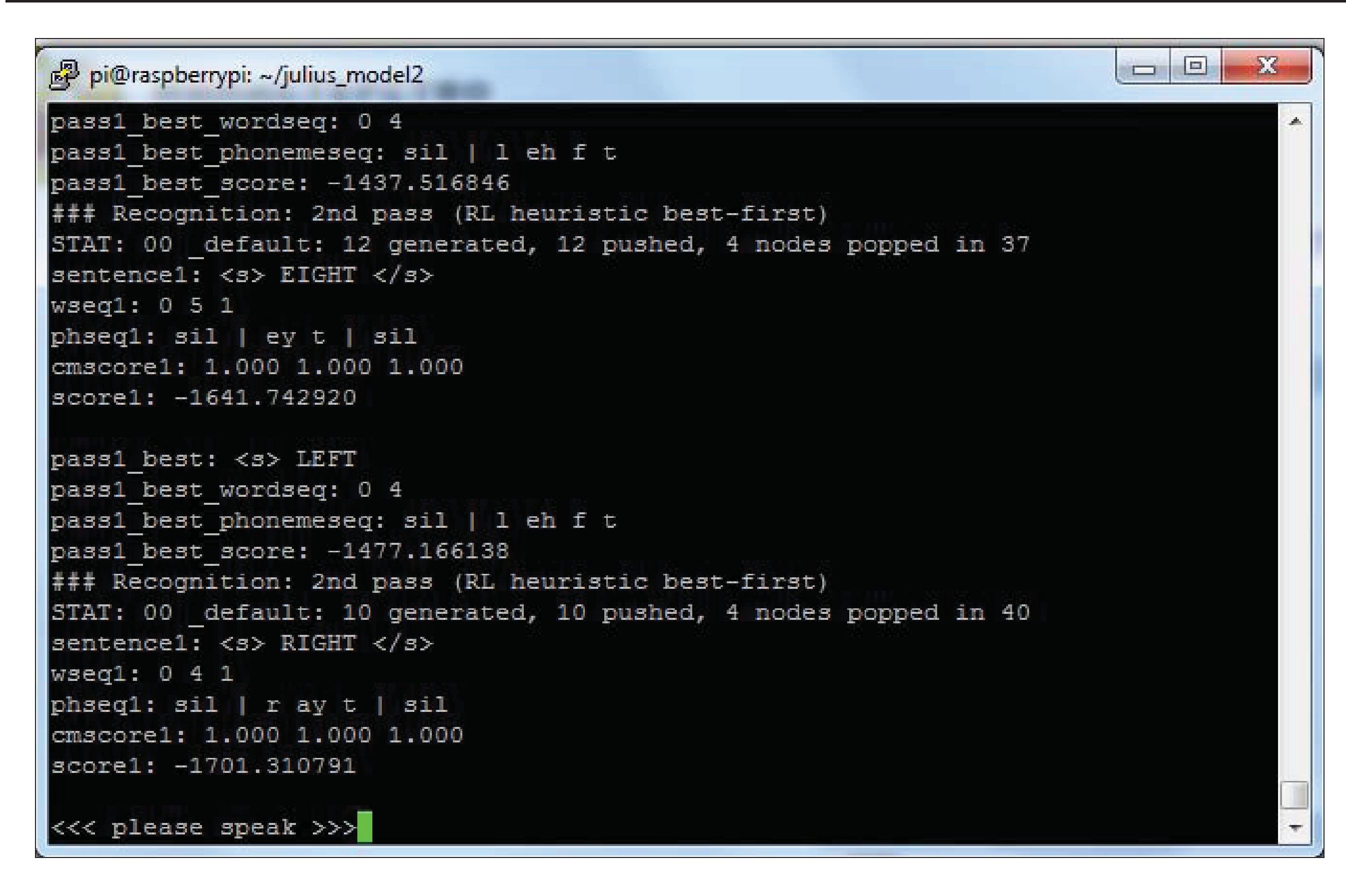


Рис. 4.33. Распознование с помощью Julius

Первый скрипт — julius_init.py — запускает скрипт распознования Julius и передает результат скрипту julius_to_text.py для выделения распознаваемого слова. Содержимое этих скриптов представлено в листингах 4.6 и 4.7 соответственно.

Листинг 4.6. Скрипт julius_init.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
import subprocess
import time
import sys

while True:
     try:
        a = os.system('julius -quiet -input mic -C
/home/pi/julius_model2/julian.jconf 2>/dev/null | ./julius_to_text.py')
        time.sleep(1)
     except KeyboardInterrupt:
        sys.exit(1)
```

Листинг 4.7. Скрипт julius_to_text.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os
```

```
import subprocess
import sys

var1=0
while var1==0:
       startstring = 'sentence1: <s> '
       endstring = ' </s>'
       line = sys.stdin.readline()
       if not line:
           pass
       else:
           end = len(line)-6
           if line[0:9] == 'sentence1':
             word = line[15:end]
             print word
```

> **ПРИМЕЧАНИЕ**
>
> Коды всех листингов этого раздела вы найдете в папке *glava_04\julius* сопровождающего книгу электронного архива (см. *приложение*).

4.13.2. Голосовое управление с использованием Google Speech API

Сервис Google Speech создавался корпорацией Google для голосового поиска. Нужно что-нибудь найти в Интернете — говорим фразу, а Google ищет. Удобно для всякого рода мобильных гаджетов. Но поиск нас не интересует. Нас интересует технология. К счастью, почти для каждого своего сервиса Google предоставляет API, которое дает возможность использовать сервис в своих приложениях. А это как раз то, что нам нужно.

Прежде всего, необходимо установить пакеты `sox` и `flac`:

```
sudo apt-get install v4l-utils sox alsa-tools alsa-oss
sudo flac
```

Сначала делаем запись голоса (произносим, например: "hello Google speech") с помощью программы `arecord` (использовался микрофон, встроенный в веб-камеру Logitech C270):

```
arecord -B --buffer-time=1000000 -f dat -r 16000 -d 4 -D plughw:1,0 send.wav
```

где:

- `-B --buffer-time=1000000` — специальный буфер (он описан в `--help`, и если его не задействовать, то будет выдана ошибка, и Google не распознает речь);
- `-f dat` — без этого параметра система делает запись 8 битов, а с этой — 16;
- `-r 16000` — запись с частотой 16 кГц;
- `-d 4` — длительность записи 4 секунды;

- `-D plughw:1,0` — наш микрофон;
- `send.wav` — файл для записи.

Далее конвертируем этот WAV-файл в формат FLAC (формат для Google Speech API, который позволяет сжимать аудио без потерь):

```
flac -f -s send.wav -o send.flac
```

Затем отправляем `POST`-запрос к сервису Google Speech API:

```
wget -q -U "Mozilla/5.0" --post-file send.flac --header "Content-Type: audio/
x-flac; rate=16000" -O - "http://www.google.com/
speech-api/v1/recognize?lang=en-us&client=chromium"
```

И получаем ответ (рис. 4.34) — сервис выдал фразу **hello Google speech** и очень высокую вероятность распознавания: 0,855.

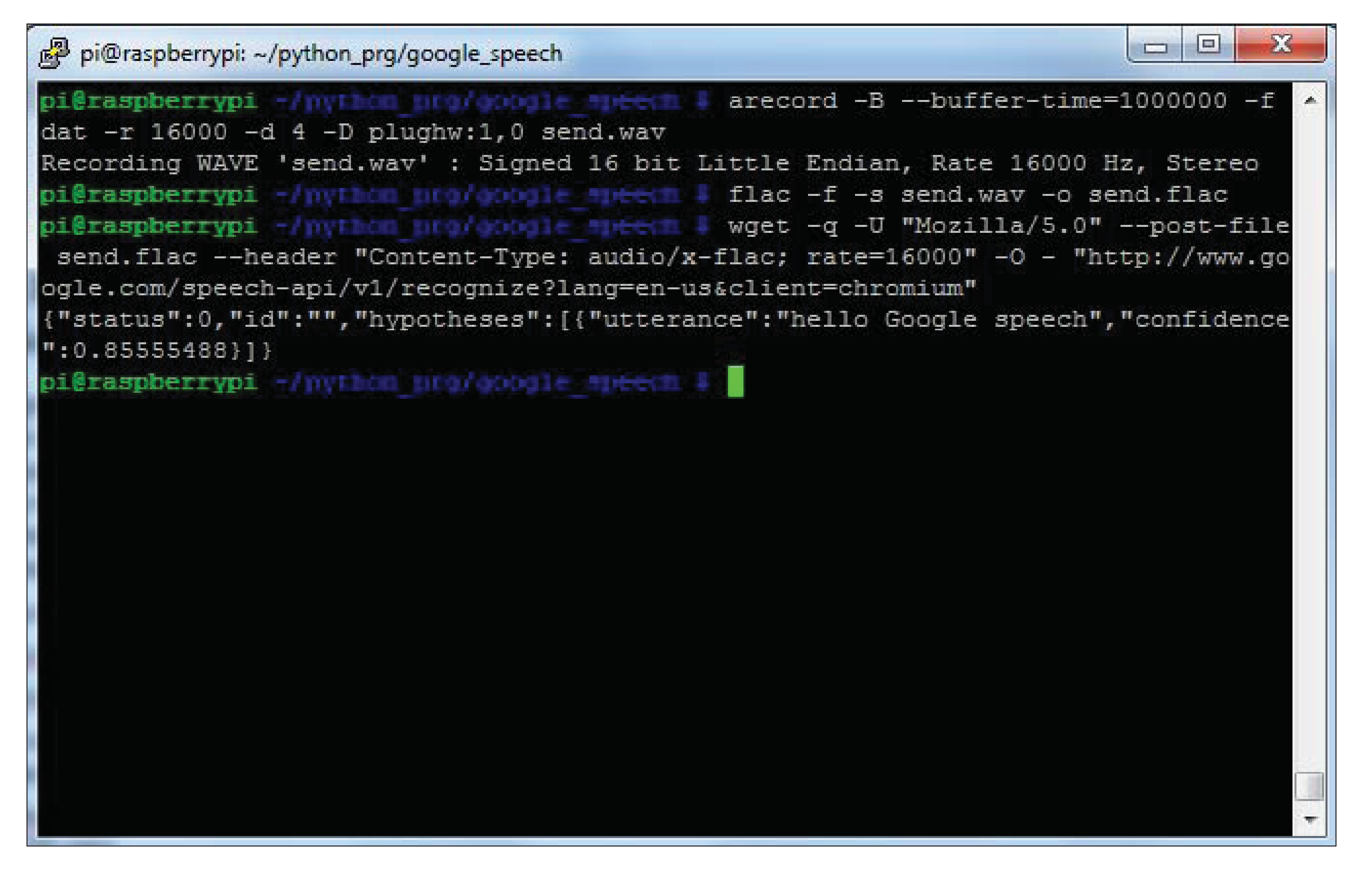


Рис. 4.34. Результат распознавания звукозаписи сервисом Google Speech API

Напишем теперь bash-скрипт, ожидающий запись с микрофона, отправляющий записанный файл на распознавание сервису Google Speech и записывающий результат в файл stt.txt.

Создаем файл скрипта stt.sh:

```
sudo touch stt.sh
```

Даем ему права на выполнение:

```
sudo chmod +x stt.sh
```

Записываем в него код, представленный в листинге 4.8.

Листинг 4.8. Скрипт stt.sh

```
export AUDIODEV="hw:0"
echo "speaking ..."
arecord -B --buffer-time=1000000 -f dat -r 16000 -d 4 -D plughw:1,0 send.wav

echo "wav - flac"
flac -f -s send.wav -o send.flac

echo "Converting Speech to Text..."
wget -q -U "Mozilla/5.0" --post-file send.flac --header "Content-Type: audio/x-flac; rate=16000" -O - "http://www.google.com/speech-api/v1/recognize?lang=en-us&client=chromium" | cut -d\" -f12  > stt.txt

echo "You Said:"
value=`cat stt.txt`
echo "$value"
```

Запускаем наш скрипт:

```
./stt.sh
```

И получаем результат (рис. 4.35).

```
pi@raspberrypi ~/python_prj/google_speech $ ./stt.sh
speaking ...
Recording WAVE 'send.wav' : Signed 16 bit Little Endian, Rate 16000 Hz, Stereo
wav - flac
Converting Speech to Text...
You Said:
hello hello hello
pi@raspberrypi ~/python_prj/google_speech $ ./stt.sh
speaking ...
Recording WAVE 'send.wav' : Signed 16 bit Little Endian, Rate 16000 Hz, Stereo
wav - flac
Converting Speech to Text...
You Said:
hello Google
pi@raspberrypi ~/python_prj/google_speech $
```

Рис. 4.35. Результат выполнения скрипта stt.sh

Примечание

Коды всех листингов этого раздела, а также файлы *send.wav* и *send.flac* вы найдете в папке *glava_04\google_speech* сопровождающего книгу электронного архива (см. *приложение*).

4.14. Raspberry Pi и ROS

ROS (Robot Operating System, операционная система для роботов) — это программный комплекс (фреймворк) для программирования роботов, предоставляющий обширный функционал. ROS была первоначально разработана в 2007 году под назва-

нием switchyard в Лаборатории искусственного интеллекта Стэнфордского университета, и ее развитие продолжается усилиями многих организаций.

Обычно при создании робота приходится реализовывать свою архитектуру, свой протокол обмена сообщениями, драйвер пульта управления, логику навигации и пр. И даже если вы сможете использовать различные готовые библиотеки для этих задач, то все равно перед вами встанет серьезная проблема — объединить их в единую систему робота. Разработчики ROS позиционируют свою систему как инструмент создания программ взаимодействия и управления роботом. ROS действительно играет роль "операционной системы", предоставляя программам управления свои интерфейсы, библиотеки и готовые приложения. ROS работает под уже готовой ОС (Ubuntu Linux), в которой реализует свой дополнительный слой абстракции для управления роботами. ROS обеспечивает стандартные службы операционной системы, такие как: аппаратная абстракция, низкоуровневый контроль устройств, реализация часто используемых функций, передача сообщений между процессами и управление пакетами.

Для ROS уже реализованы драйверы, позволяющие единым образом работать со многими устройствами: джойстиками, GPS, камерами, лазерными дальномерами и пр.

ROS имеет две основные ипостаси: именно операционной системы, как только что описано, и ros-pkg — набора поддерживаемых пользователями пакетов (которые называются *стеками*), реализующих различные функции робототехники. ROS уже содержит вспомогательные библиотеки и приложения для роботов: преобразование систем координат, утилиты для визуализации данных, распознавания объектов, стек навигации и многое другое.

ROS основан на архитектуре графов, где обработка данных происходит в узлах, которые могут получать и передавать сообщения (структурированные данные) между собой. Комбинируя готовые узлы ROS и, по необходимости, дописывая собственные, можно существенно сократить время разработки и позволить себе сконцентрироваться только на тех задачах, которые действительно нужно решить.

На данный момент уже много роботов работает под управлением ROS. Вот неполный список: PR2, TurtleBot, PR1, HERB, STAIR I и II, Nao, Husky A200, iRobot Create, Lego Mindstorms NXT.

ROS распространяется в соответствии с условиями BSD-лицензии и с открытым исходным кодом. ROS бесплатна для использования как в исследовательских, так и в коммерческих целях. Пакеты из ros-pkg распространяются на условиях различных открытых лицензий.

4.14.1. Установка ROS-дистрибутива Hydro на Raspberry Pi

Рассмотрим установку последнего дистрибутива ROS — Hydro — на компьютер Raspberry Pi (система Raspbian). Процесс установки продлится нескольких часов, имейте это в виду.

Сначала подключим репозиторий:

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu raring main" >
/etc/apt/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O - | sudo apt-key add –
sudo apt-get update
sudo apt-get upgrade
```

Затем установим зависимости:

```
sudo apt-get install python-pip
sudo pip install rosdistro
sudo pip install wstool
```

Установим пакет `setuptools`:

```
mkdir  Downloads
cd ~/Downloads
wget https://pypi.python.org/packages/source/s/setuptools/setuptools-
1.1.6.tar.gz
tar xvf setuptools-1.1.6.tar.gz
cd setuptools-1.1.6
sudo python setup.py install
```

Установим пакет `stdeb`:

```
sudo apt-get install python-stdeb
```

Установим зависимости:

```
sudo pip install rosdep
sudo pip install rosinstall-generator
sudo pip install wstool
pypi-install rospkg
sudo apt-get install python-rosdep python-rosinstall-generator build-essential
```

Теперь переходим собственно к установке Hydro с ее исходной страницы. Вот именно эти операции несколько часов и займут.

```
mkdir ~/ros_catkin_ws
cd ~/ros_catkin_ws
rosinstall_generator ros_comm --rosdistro hydro --deps --wet-only > hydro-
ros_comm-wet.rosinstall
wstool init -j8 src hydro-ros_comm-wet.rosinstall
sudo rosdep init
rosdep update
rosdep install  --from-paths src --ignore-src --rosdistro hydro -y --
os=debian:wheezy
```

Здесь выскакивает ошибка... Она связана с отсутствием пакета `SBCL`, используемого пакетом `roslisp`. Для выхода из этой ситуации удалим пакет `roslisp`:

```
cd src
wstool rm roslisp
rm -rf roslisp
```

```
cd ..
rosdep install  --from-paths src --ignore-src --rosdistro hydro -y --
os=debian:wheezy
```

Теперь все нормально.

Построим рабочее пространство `catkin`:

```
./src/catkin/bin/catkin_make_isolated -install
```

Сделаем, чтобы рабочее пространство ROS инициализировалось при входе пользователя в терминал:

```
echo "source ~/ros_catkin_ws/install_isolated/setup.bash" >> .bashrc
source .bashrc
```

Для проверки запускаем команду `roscore` (рис. 4.36).

```
roscore http://raspberrypi:11311/
pi@raspberrypi ~ $ source .bashrc
pi@raspberrypi ~ $ roscore
... logging to /home/pi/.ros/log/31f8cd12-6c7c-11e3-937c-b827eb83a32c/roslaunch-
raspberrypi-14929.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://raspberrypi:33100/
ros_comm version 1.9.50


SUMMARY
========

PARAMETERS
 * /rosdistro
 * /rosversion

NODES

auto-starting new master
process[master]: started with pid [14948]
ROS_MASTER_URI=http://raspberrypi:11311/

setting /run_id to 31f8cd12-6c7c-11e3-937c-b827eb83a32c
process[rosout-1]: started with pid [14961]
started core service [/rosout]
```

Рис. 4.36. Запуск команды `roscore`

Что ж, все работает. Теперь можно приступать к созданию своих проектов в ROS.

> ***Примечание***
>
> Документацию по ROS можно найти на ее официальном сайте: **http://www.rps.org**. Очень хорошие русские руководства по ROS находятся здесь: **http://robocraft.ru/page/robotics/#ROS**.

4.14.2. Создание тестового проекта

Создадим на языке Python проект простых узлов: publisher (узел, публикующий сообщения в какую-нибудь тему в ROS) и subscriber (узел-слушатель сообщений из определенной темы ROS).

Для начала создадим проект в `catkin` (новая система построения пакетов в ROS, призванная заменить систему построения пакетов `rosdep`).

Для создания в `catkin` нового пакета `beginner_tutorials` выполняем следующие команды:

```
cd ~/ros_catkin_ws/src
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```

При создании пакета сразу указываем зависимости. Система сообщает о создании нового пакета, и в каталоге src создается папка beginner_tutorials с необходимым набором файлов (рис. 4.37).

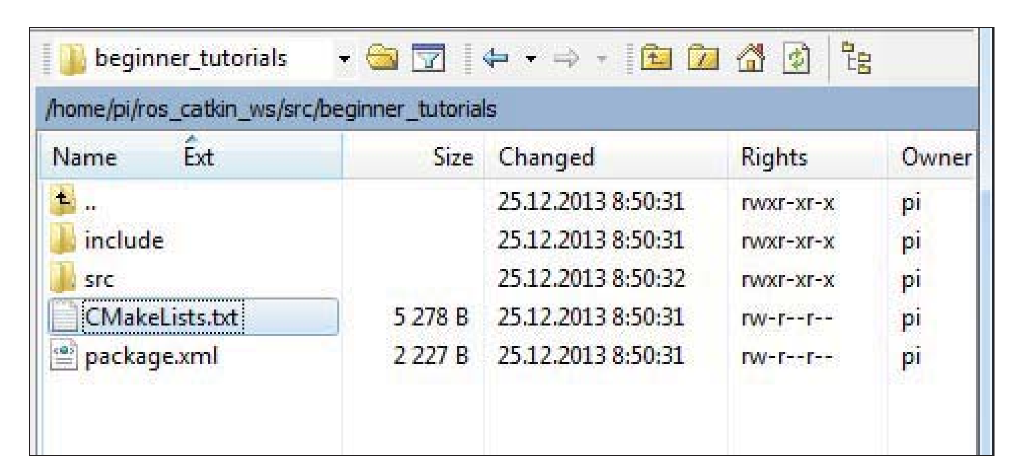


Рис. 4.37. Содержимое каталога созданного пакета beginner_tutorials

Внесем небольшие изменения в файл package.xml. В теги <name>, <version>, <description>, <maintainer>, <license>, <url>, <author> вносится информация о названии, версии, авторе, лицензии пакета. В теге <depends> указаны зависимости пакета. Все эти зависимости будут доступны в сборке и во время выполнения, поэтому мы добавим тег <run_depend> для каждой из них:

```
<run_depend>roscpp</run_depend>
<run_depend>rospy</run_depend>
<run_depend>std_msgs</run_depend>
```

Далее переходим в наш пакет:

```
source devel/setup.bash
roscd beginner_tutorials
```

Затем пишем код. Создаем каталог scripts и в ней файл taker.py. Заносим в него код (листинг 4.9) и даем ему права на выполнение:

```
mkdir scripts
cd scripts
```

```
touch talker.py
sudo nano talker.py
sudo chmod +x talker.py
```

Листинг 4.9. Файл taker.py

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String)
    rospy.init_node('talker')
    while not rospy.is_shutdown():
        str = "hello world %s" % rospy.get_time()
        rospy.loginfo(str)
        pub.publish(String(str))
        rospy.sleep(1.0)

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

Скрипт talker.py каждую секунду отправляет сообщение типа `String` в тему `chatter`. Содержимое сообщения: "hello world" — с добавлением системного времени.

Создаем скрипт subscriber (слушателя) — listener.py:

```
cd scripts
touch listener.py
sudo nano listener.py
sudo chmod +x listener.py
```

Содержимое для файла listener.py приведено в листинге 4.10.

Листинг 4.10. Файл listener.py

```
#!/usr/bin/env python
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo(rospy.get_name() + ": I heard %s" % data.data)

def listener():
    rospy.init_node('listener', anonymous=True)
```

```
    rospy.Subscriber("chatter", String, callback)
    rospy.spin()

if __name__ == '__main__':
    listener()
```

Обратите внимание — мы применяем CMake[1] в качестве нашей системы сборки. Да, вы должны использовать ее даже для узлов Python. Это делается для уверенности, что создается автоматически сгенерированный Python-код для сообщений и сервисов.

```
cd ~/ros_catkin_ws
catkin_make
```

Теперь проверим работу нашего пакета `beginner_tutorials`.

Запускаем в первом терминале:

```
roscore
```

Во втором терминале запускаем на выполнение скрипт talker.py из пакета `beginner_tutorials`:

```
rosrun beginner_tutorials talker.py
```

И наблюдаем ежесекундную выдачу сообщений скриптом talker.py в тему `chatter` (рис. 4.38).

```
roscore http://raspberrypi:11311/
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://raspberrypi:57128/
ros_comm version 1.9.50


SUMMARY
========

PARAMETERS
 * /rosdistro
 * /rosversion

NODES

auto-starting new master
process[master]: started with pid [6129]
ROS_MASTER_URI=http://raspberrypi:11311/

setting /run_id to 47975b9c-6d36-11e3-8662-b827eb83a32c
process[rosout-1]: started with pid [6142]
started core service [/rosout]
```

```
pi@raspberrypi: ~/ros_catkin_ws
[ 88%] Built target throttle
[ 90%] Built target topic_tools_generate_messages_lisp
[ 93%] Built target topic_tools_generate_messages_py
[ 93%] Built target topic_tools_generate_messages
[ 98%] Built target rosbag
[ 98%] Built target play
[100%] Built target record
pi@raspberrypi ~/ros_catkin_ws $ rosrun beginner_tutorials talker.py
[INFO] [WallTime: 1387956613.369053] hello world 1387956613.37
[INFO] [WallTime: 1387956614.379886] hello world 1387956614.38
[INFO] [WallTime: 1387956615.392854] hello world 1387956615.39
[INFO] [WallTime: 1387956616.402575] hello world 1387956616.4
[INFO] [WallTime: 1387956617.435833] hello world 1387956617.43
[INFO] [WallTime: 1387956618.447589] hello world 1387956618.45
[INFO] [WallTime: 1387956619.459849] hello world 1387956619.46
[INFO] [WallTime: 1387956620.468537] hello world 1387956620.47
[INFO] [WallTime: 1387956621.481500] hello world 1387956621.48
[INFO] [WallTime: 1387956622.499989] hello world 1387956622.5
[INFO] [WallTime: 1387956623.513397] hello world 1387956623.51
[INFO] [WallTime: 1387956624.522792] hello world 1387956624.52
[INFO] [WallTime: 1387956625.534809] hello world 1387956625.53
[INFO] [WallTime: 1387956626.543821] hello world 1387956626.54
[INFO] [WallTime: 1387956627.564001] hello world 1387956627.56
```

Рис. 4.38. Публикация сообщений скриптом talker.py в тему `chatter`

В третьем терминале запускаем на выполнение слушатель — скрипт listener.py:

```
rosrun beginner_tutorials listener.py
```

[1] CMake (от англ. *cross platform make*) — кроссплатформенная система автоматизации сборки программного обеспечения из исходного кода.

И видим получение сообщений скриптом listener.py сообщений из темы `chatter` (рис. 4.39).

Командами `rostopic list` и `rosnode list` смотрим список тем и узлов (рис. 4.40).

```
pi@raspberrypi: ~/ros_catkin_ws
pi@raspberrypi ~/ros_catkin_ws $ rosrun beginner_tutorials listener.py
[INFO] [WallTime: 1387957991.574965] /listener_6789_1387957990281: I heard hello
 world 1387957991.56
[INFO] [WallTime: 1387957992.599518] /listener_6789_1387957990281: I heard hello
 world 1387957992.59
[INFO] [WallTime: 1387957993.617896] /listener_6789_1387957990281: I heard hello
 world 1387957993.61
[INFO] [WallTime: 1387957994.651528] /listener_6789_1387957990281: I heard hello
 world 1387957994.63
[INFO] [WallTime: 1387957995.672972] /listener_6789_1387957990281: I heard hello
 world 1387957995.66
```

Рис. 4.39. Получение сообщений скриптом listener.py из темы chatter

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ rosnode list
/listener_6789_1387957990281
/rosout
/talker
pi@raspberrypi ~ $ rostopic list
/chatter
/rosout
/rosout_agg
pi@raspberrypi ~ $
```

Рис. 4.40. Просмотр списка узлов и тем

4.15. Raspberry Pi и OpenCV

OpenCV (Open Computer Vision, библиотека "компьютерного зрения" с открытым исходным кодом) — предоставляет нам набор типов данных и численных алгоритмов для обработки изображений алгоритмами "компьютерного зрения". OpenCV написана на языке высокого уровня (C/C++) и содержит алгоритмы для интерпретации изображений, калибровки камеры по эталону, устранения оптических иска-

жений, определения сходства, анализа перемещения объекта, определения формы объекта и слежения за объектом, 3D-реконструкции, сегментации объекта, распознавания жестов и пр.

Эта библиотека очень популярна из-за своей открытости и возможности бесплатно использовать ее как в учебных, так и коммерческих целях.

Для установки пакета `opencv` на устройство Raspberry Pi под управлением дистрибутива Raspbian выполним следующую команду:

```
sudo apt-get install libopencv-dev python-opencv
```

По этой команде будет установлена версия 2.3.1 пакета `opencv`. А пакет `python-opencv` пригодится нам для написания программ на языке Python с использованием `opencv`.

Итак, напишем первую программу на Python с использованием библиотеки `opencv`.

Создадим для наших программ каталог prg-opencv\helloworld и в нем файл hello.py:

```
cd ~
mkdir prg-opencv
cd prg-opencv
mkdir helloworld
touch hello.py
```

И занесем в файл hello.py содержимое листинга 4.11.

Листинг 4.11. Файл hello.py

```
#!/usr/bin/python

import cv2
import sys

PATH="/home/pi/prg-opencv/helloworld/"

if len(sys.argv) != 2:
    sys.exit("Expecting a single image file argument")
filename = sys.argv[1]

image = cv2.imread((PATH+filename),0)
print image.shape
image_small = cv2.resize(image, (600, 400))
textColor = (0, 0, 255)  # red
cv2.putText(image_small, "Hello World!!!", (200, 200),
            cv2.FONT_HERSHEY_PLAIN, 3.0, textColor,
            thickness=4, linetype=cv2.CV_AA)
cv2.imshow('Hello World GUI', image_small)
cv2.waitKey()
cv2.destroyAllWindows()
```

Запускаем скрипт hello.py на выполнение:

```
python /home/pi/prg-opencv/helloworld/hello.py img1.png
```

Результат выполнения представлен на рис. 4.41.

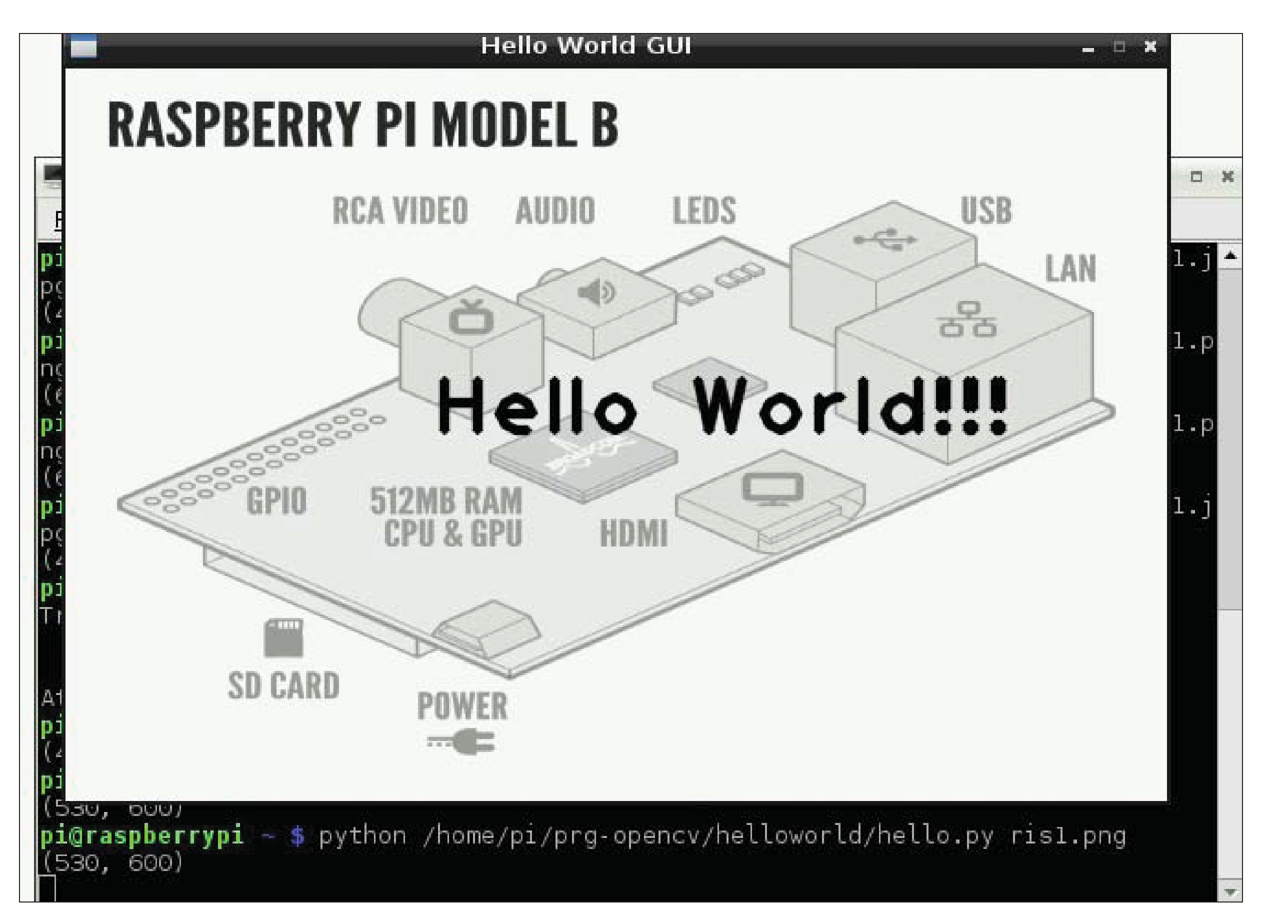


Рис. 4.41. "Hello World" для `python-opencv`

4.15.1. Получение в OpenCV изображения с камеры

Распознавание образов — это одна из самых интересных областей компьютерной графики. В то же время это и одна из самых сложных ее областей. К счастью, нам не придется разбираться в сложностях математической теории. Распознавание лиц осуществляется по так называемым *признакам Хаара*. "Эталон" лица уже давно создан и хорошо изучен, находится он в файле haarcascade_frontalface_alt.xml. Файл содержит так называемую *модель лица*, а именно "идеальное" лицо в виде примитивов Хаара (рис. 4.42).

"Идеальное" лицо представляет собой бинаризированную диаграмму яркостей. У любого лица посередине светло (нос), по краям оно уходит в темное, а потом снова на светлое (щеки). Таких признаков у лица много, они и приведены в файле haarcascade_frontalface_alt.xml.

Содержимое скрипта facedetect.pi представлено в листинге 4.12.

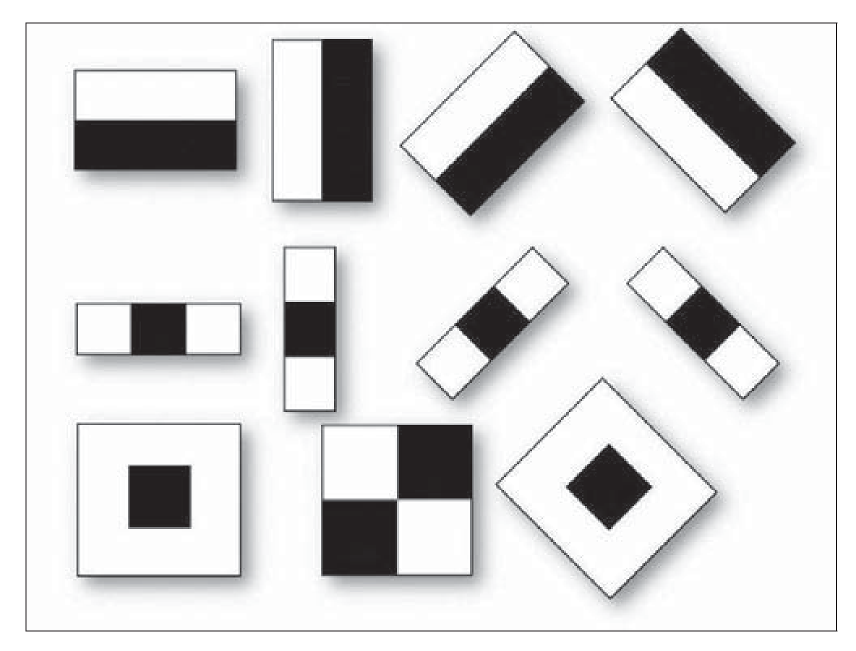


Рис. 4.42. "Идеальное" лицо в виде примитивов Хаара

Листинг 4.12. Скрипт facedetect.pi

```
#!/usr/bin/python
import sys
import cv2.cv as cv
from optparse import OptionParser

# default parameters (scale_factor=2, min_neighbors=3, flags=0)
# images the settings are:
# scale_factor=1.2, min_neighbors=2, flags=CV_HAAR_DO_CANNY_PRUNING,
# min_size=<minimum possible face size

min_size = (20, 20)
image_scale = 2
haar_scale = 1.2
min_neighbors = 2
haar_flags = 0

def detect_and_draw(img, cascade):
    # allocate temporary images
    gray = cv.CreateImage((img.width,img.height), 8, 1)
    small_img = cv.CreateImage((cv.Round(img.width / image_scale),
                            cv.Round (img.height / image_scale)), 8, 1)

    # convert color input image to grayscale
    cv.CvtColor(img, gray, cv.CV_BGR2GRAY)

    # scale input image for faster processing
    cv.Resize(gray, small_img, cv.CV_INTER_LINEAR)
```

```
    cv.EqualizeHist(small_img, small_img)

    if(cascade):
        t = cv.GetTickCount()
        faces = cv.HaarDetectObjects(small_img, cascade,
cv.CreateMemStorage(0),haar_scale, min_neighbors, haar_flags, min_size)
        t = cv.GetTickCount() - t
        print "detection time = %gms" % (t/(cv.GetTickFrequency()*1000.))
        if faces:
            for ((x, y, w, h), n) in faces:
                pt1 = (int(x * image_scale), int(y * image_scale))
                pt2 = (int((x + w) * image_scale), int((y + h) * image_scale))
                cv.Rectangle(img, pt1, pt2, cv.RGB(255, 0, 0), 3, 8, 0)
    cv.ShowImage("result", img)

if __name__ == '__main__':
    parser = OptionParser(usage = "usage: %prog [options]
[filename|camera_index]")
    parser.add_option("-c", "--cascade", action="store", dest="cascade",
type="str", help="Haar cascade file, default %default", default =
"../data/haarcascades/haarcascade_frontalface_alt.xml")
    (options, args) = parser.parse_args()
    cascade = cv.Load(options.cascade)

    if len(args) != 1:
        parser.print_help()
        sys.exit(1)

    input_name = args[0]
    if input_name.isdigit():
        capture = cv.CreateCameraCapture(int(input_name))
    else:
        capture = None

    cv.NamedWindow("result", 1)

    width = 320 #leave None for auto-detection
    height = 240 #leave None for auto-detection

    if width is None:
       width = int(cv.GetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_WIDTH))
    else:
       cv.SetCaptureProperty(capture,cv.CV_CAP_PROP_FRAME_WIDTH,width)

    if height is None:
       height = int(cv.GetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_HEIGHT))
    else:
       cv.SetCaptureProperty(capture,cv.CV_CAP_PROP_FRAME_HEIGHT,height)
```

```
if capture:
    frame_copy = None
    while True:
        frame = cv.QueryFrame(capture)
        if not frame:
            cv.WaitKey(0)
            break
        if not frame_copy:
            frame_copy = cv.CreateImage((frame.width,frame.height),
                        cv.IPL_DEPTH_8U, frame.nChannels)
        if frame.origin == cv.IPL_ORIGIN_TL:
            cv.Copy(frame, frame_copy)
        else:
            cv.Flip(frame, frame_copy, 0)

        detect_and_draw(frame_copy, cascade)

        if cv.WaitKey(10) >= 0:
            break
else:
    image = cv.LoadImage(input_name, 1)
    detect_and_draw(image, cascade)
    cv.WaitKey(0)

cv.DestroyWindow("result")
```

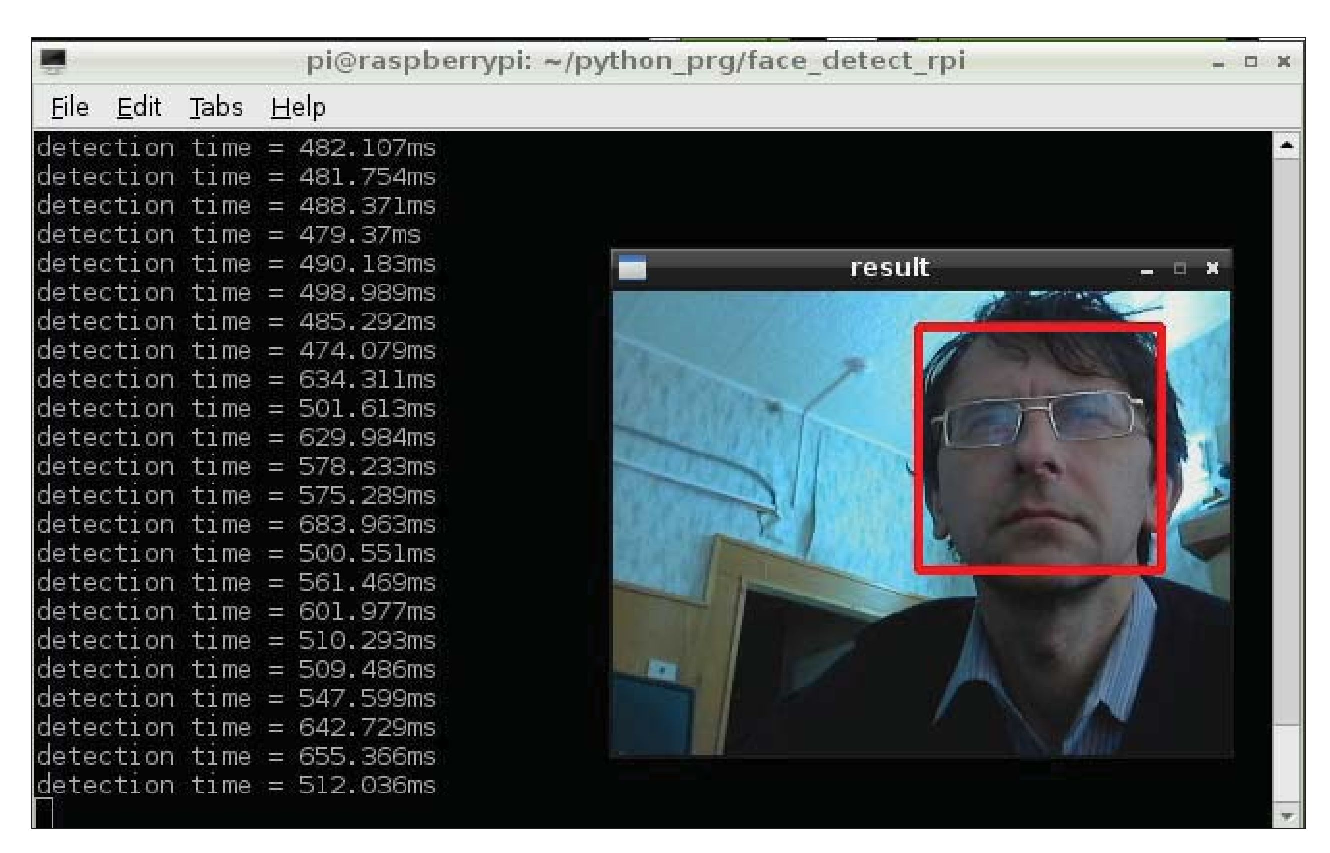


Рис. 4.43. Детектирование лица на видеопотоке с веб-камеры

Для запуска этого скрипта выполните команду:

```
python facedetect.py --cascade=face.xml 0
```

где:

- `face.xml` — шаблон модели лица;
- `0` — номер камеры, с которой берем изображение.

Результат выполнения скрипта — красный прямоугольник вокруг опознанного лица (рис. 4.43).

> **ПРИМЕЧАНИЕ**
>
> Код скрипта *facedetect.pi*, а также файл *face.xml* вы найдете в папке *glava_04\face_detect_rpi* сопровождающего книгу электронного архива (см. *приложение*).

4.16. Подключение платы Arduino

Arduino — аппаратная вычислительная платформа, основными компонентами которой являются простая плата ввода/вывода и среда разработки на языке Processing/Wiring. Arduino может использоваться как для создания автономных интерактивных объектов, так и подключаться к программному обеспечению, выполняемому на компьютере. Короче говоря, это "золотое дно" для новичков, поскольку на основе Arduino можно разрабатывать конструкции простого и среднего уровня за весьма короткие сроки. Arduino является и великолепной микроконтроллерной платформой для отладки и прототипирования с огромным количеством готовых проектов с открытым исходным кодом, учебных материалов, форумов и пр., что очень важно для всех при изучении встраиваемых систем.

Используя простую интегрированную среду разработки и код на C++-подобном языке, USB-кабель и несколько пассивных компонентов, можно в считанные секунды заставить мигать светодиод или за несколько минут организовать обмен данными с ПК, не имея серьезного опыта в электронике. Нет надобности и в создании законченных плат и модулей — разработчик может использовать готовые платы расширения (которых очень много) или просто напрямую подключить к Arduino необходимые элементы. Создано для Arduino и множество библиотек, содержащих код, работающий с различными устройствами.

Связка Raspberry Pi с Arduino является очень перспективной, т. к. позволяет добавить к контроллеру Arduino вычислительные возможности Raspberry Pi.

Давайте сразу установим Arduino IDE на Raspberry Pi (система Raspbian):

```
sudo apt-get update
sudo apt-get install arduino
```

На момент написания этой книги устанавливается версия 1.0.1. Набираем в терминале:

```
arduino
```

и перед нами среда программирования Arduino IDE (рис. 4.44).

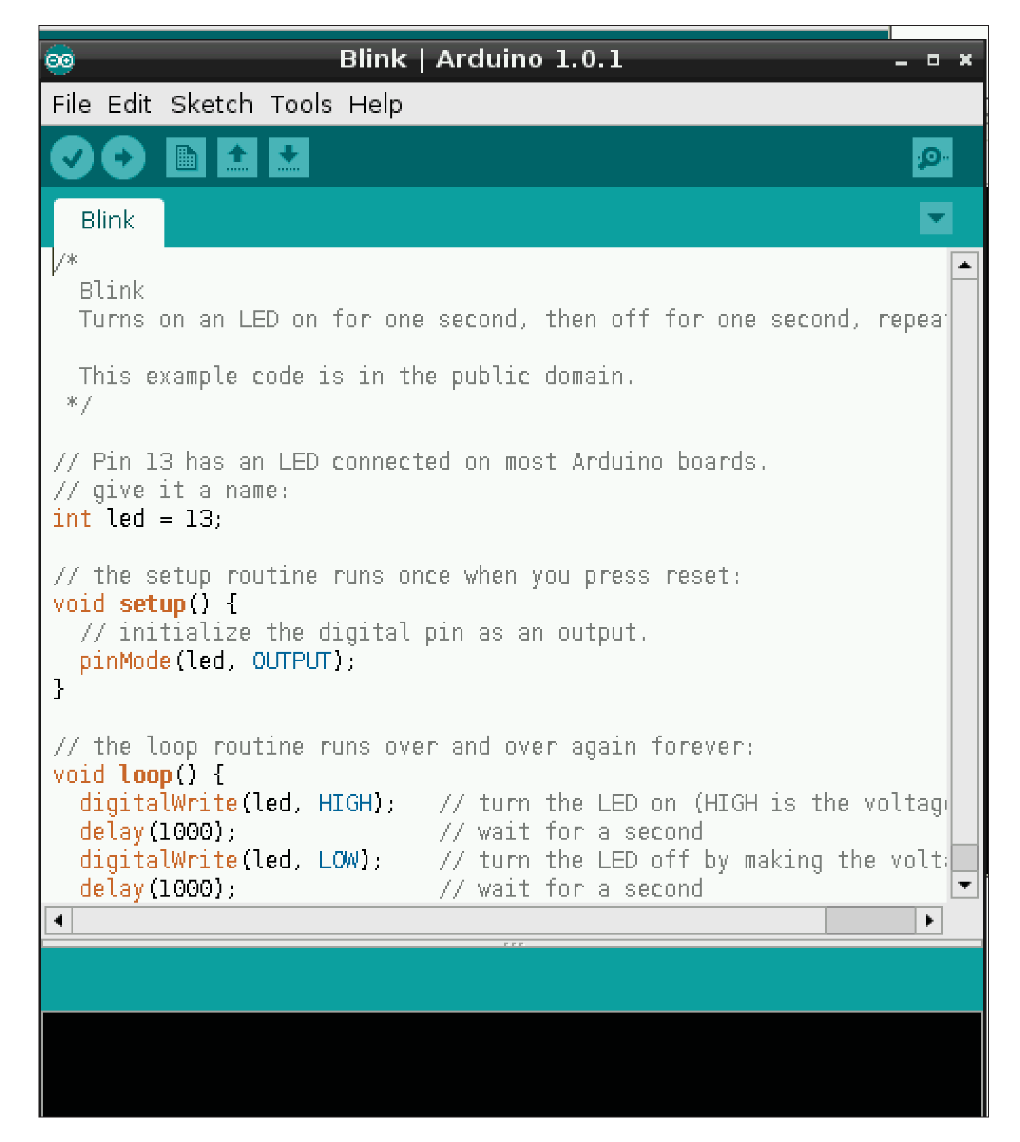


Рис. 4.44. Среда программирования Arduino IDE

Среда разработки Arduino IDE основана на языке программирования Processing и спроектирована для применения новичками, близко не знакомыми с разработкой программного обеспечения. Язык программирования аналогичен используемому в проекте Wiring. Строго говоря, это C++, дополненный некоторыми библиотеками. Программы обрабатываются препроцессором, а затем компилируются с помощью AVR-GCC.

Теперь необходимо подключить плату Arduino к USB-порту Raspberry Pi, выставить настройки (рис. 4.45 и 4.46) и можно приступать к программированию.

4.16.1. Отправка данных на сайт Народного мониторинга связкой Raspberry Pi + Arduino

Народный мониторинг (**http://www.narodmon.ru**) — проект по сбору и отображению на карте мира показаний (температура, давление, влажность и т. п.) практически в реальном времени по фактическому состоянию (а не на основе прогнозов) от различных датчиков среды, установленных как на улице для публичного доступа, так и в помещении для приватного.

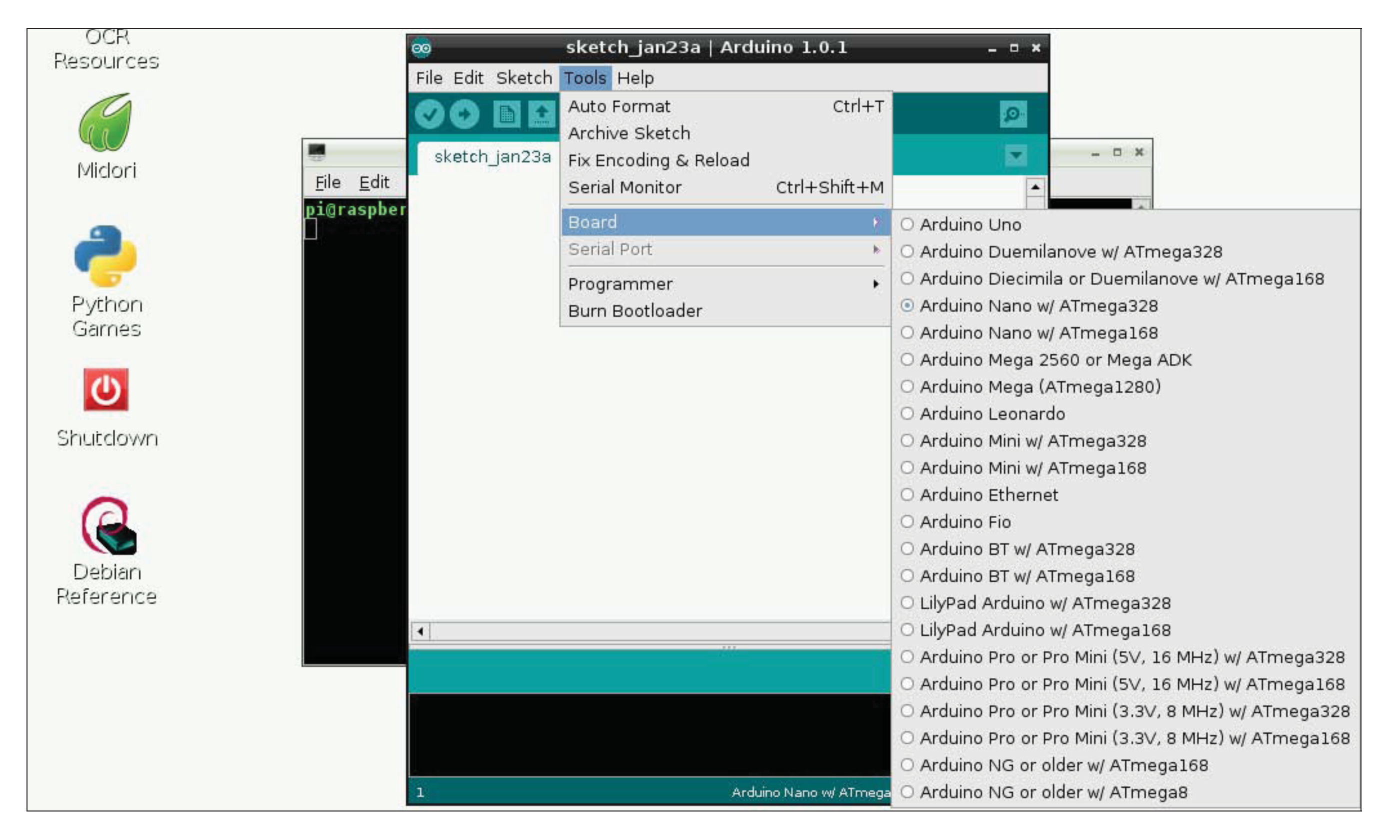


Рис. 4.45. Настройка типа используемой платы Arduino

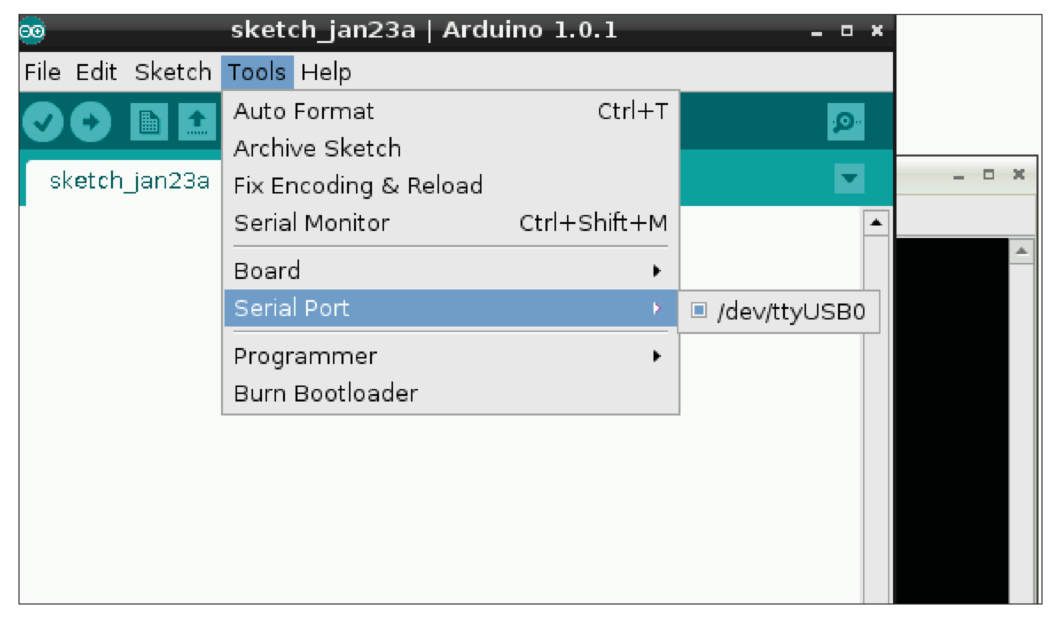


Рис. 4.46. Настройка порта подключения

Ограничения бесплатного использования сервиса:

- для передачи показаний: одно устройство на IP-адрес, интервал приема от 5 минут, хранение показаний 30 дней;
- для камер: размер картинки до 640 пикселов, интервал скачивания от 10 минут, хранение 24 часа.

Для сбора данных была собрана метеостанция на Arduino (собирает показания с датчиков DS18B20, DHT11, BMP085). Raspberry Pi по командам планировщика cron получает данные из Arduino и отправляет на сайт Народного мониторинга.

В листинге 4.13 представлен скетч для сбора данных со следующих датчиков:

- ds18b20 — цифровой датчик температуры;
- dht11 — модуль для измерения относительной влажности воздуха;
- bmp085 — блок для измерения атмосферного давления воздуха.

Обратите внимание на размер кода скетча — результат использования распространенных библиотек для Arduino.

Листинг 4.13

```
#include "SPI.h"
#include "OneWire.h"
#include "Wire.h"
#include "DHT.h"
#include "BMP085.h"

BMP085 dps = BMP085();
long Pressure085 = 0;
#define DHTTYPE DHT11   // DHT 11
DHT dht(8, DHTTYPE);
OneWire  ds(7);  // on pin 7
byte my_addr[8]={0x28,0xB0,0x16,0xB8,3,0,0,0x90};

void setup() {
  Serial.begin(9600);
  Wire.begin();
  dps.init();
}

void loop ()
 {
  if (Serial.available()>0)
    {
    if(Serial.read()=='1')
      {
      int Temp=get_temp();
      Serial.print("#28B016B803000090#");
      Serial.print(Temp/16);
      Serial.print(".");
      Serial.print(abs((Temp%16)*100)/16);
      Serial.print("&");
      float h = dht.readHumidity();
      Serial.print("#29B016B803000090#");
      Serial.print(h);
      Serial.print("&");
      Serial.print("#30B016B803000090#");
```

```
      Serial.print(dps.getPressure(&Pressure085));
      Serial.println();
     }
   }
 }
// получение температуры датчика
int get_temp()
 {
byte i;
byte present = 0;
byte data[12];
byte addr[8];
int Temp;


  ds.reset();
  ds.select(my_addr);
  ds.write(0x44,1);
  delay(1000);
  present = ds.reset();
  ds.select(my_addr);
  ds.write(0xBE);          // Read Scratchpad
  for ( i = 0; i < 9; i++) {
    data[i] = ds.read();
  }
  Temp=(data[1]<<8)+data[0];
  Temp=Temp;
  return Temp;
 }
```

На рис. 4.47 представлена схема подключения датчиков к Arduino.

Сборка Arduino прослушивает последовательный порт, по которому она подключена к компьютеру Raspberry Pi. При получении из последовательного порта символа '1' происходит съем показаний с датчиков и отправка сообщения в последовательный порт в виде:

```
#2881C4BA0200003B#TEMP&#2981C4BA0200003B#HUMIDITY&#3081C4BA0200003B#PRESSURE
```

где:

- `#2881C4BA0200003B`, `#2881C4BA0200003B`, `#2881C4BA0200003B` — идентификаторы датчиков в системе Народного мониторинга;
- `TEMP`, `HUMIDITY`, `PRESSURE` — показания датчиков.

Идентификаторы датчиков для Народного мониторинга берутся произвольные. Так, для датчика температуры я брал уникальный идентификатор 1-Wire датчика ds18b20, для датчиков влажности и давления изменял первые 2 цифры идентификатора датчика температуры.

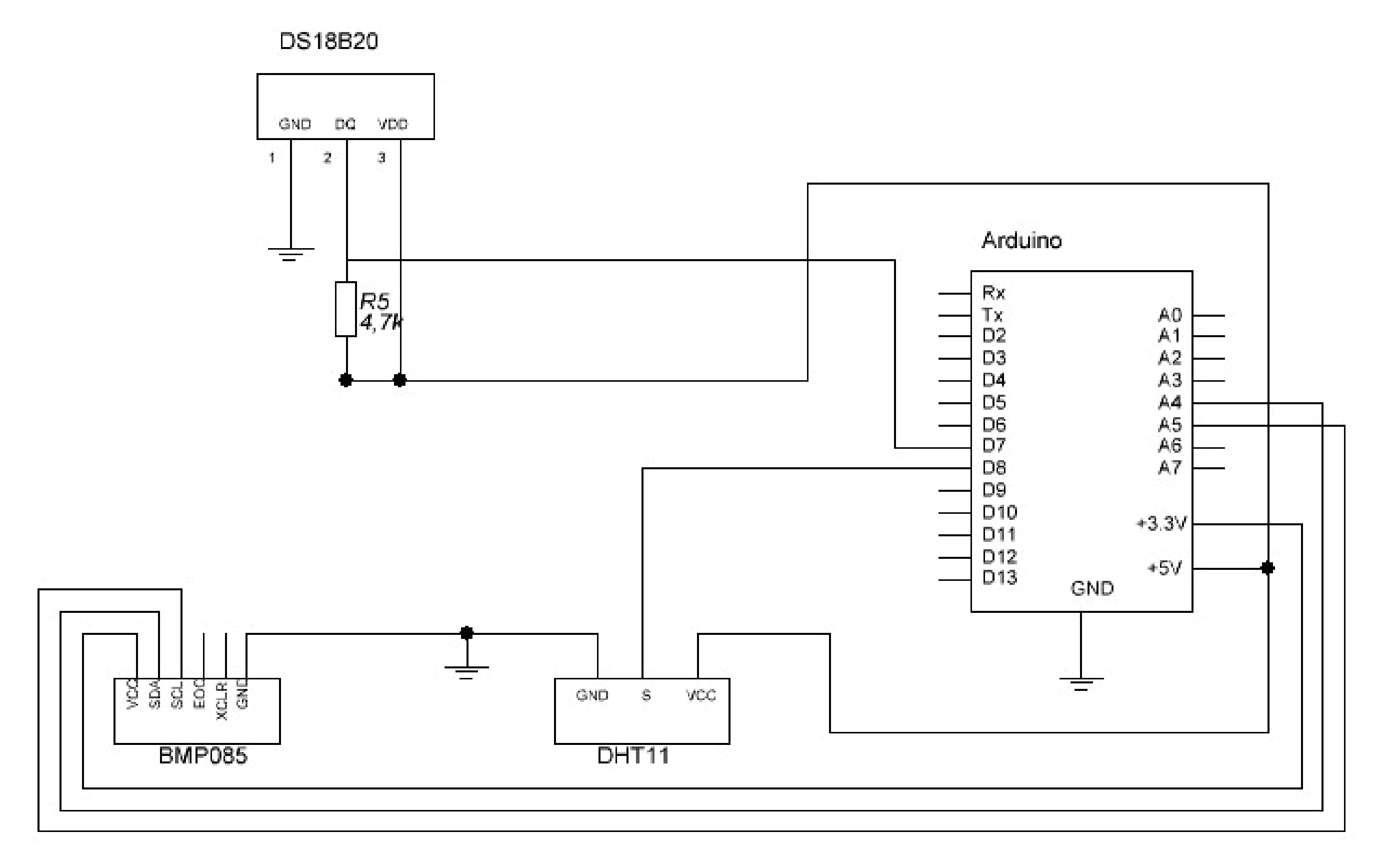


Рис. 4.47. Схема подключения датчиков к Arduino

Теперь пишем скрипт на PHP для отправки данных на сайт Народного мониторинга. Скрипт соединяется с последовательным портом, отправляет в порт символ '1' и получает из последовательного порта строку с данными датчиков. И затем после небольшого преобразования строки отправляет ее на сервер **narodmon.ru**. Задержка в 5 секунд перед отправкой данных в порт связана с тем, что при открытии порта Arduino плата перезагружается. Содержимое скрипта cron_narodmon.php представлено в листинге 4.14.

Листинг 4.14. Скрипт cron_narodmon.php

```
 <?php

define('SERIAL_DEVICE', '/dev/ttyACM0');
$fp = fopen(SERIAL_DEVICE, "w+");
if( !$fp) {
 die("can\'t open " . SERIAL_DEVICE);
}
else
  print "open port - ok\n";
  sleep(5);

 if( fwrite($fp, "1" ))  {
  print "OK\n\n";
 }
```

```
 else {
  print "FAILED!!!\n\n";
 }
$cc="";
$x=true;
while($x==true){
   $c=fread($fp,1);
   if($c=="\n")
      $x=false;
   $cc=$cc.$c;
   }

$sdata="#B8:27:EB:83:A3:2C\n".str_replace("&","\n",$cc)."##";
print $sdata;
fclose($fp);

$fs = @fsockopen("tcp://narodmon.ru", 8283, $errno, $errstr);
if(!$fs) exit("ERROR(".$errno."): ".$errstr);
fwrite($fs, $sdata);
fclose($fs);
?>
```

Чтобы зарегистрировать датчики в системе Народного мониторинга, необходимо предварительно отправить данные на сайт.

Запускаем скрипт cron_narodmon.php:

```
php-cgi /home/pi/cron_narodmon.php
```

Заходим в свой профиль на сайте **narodmon.ru** и добавляем датчики (рис. 4.48).

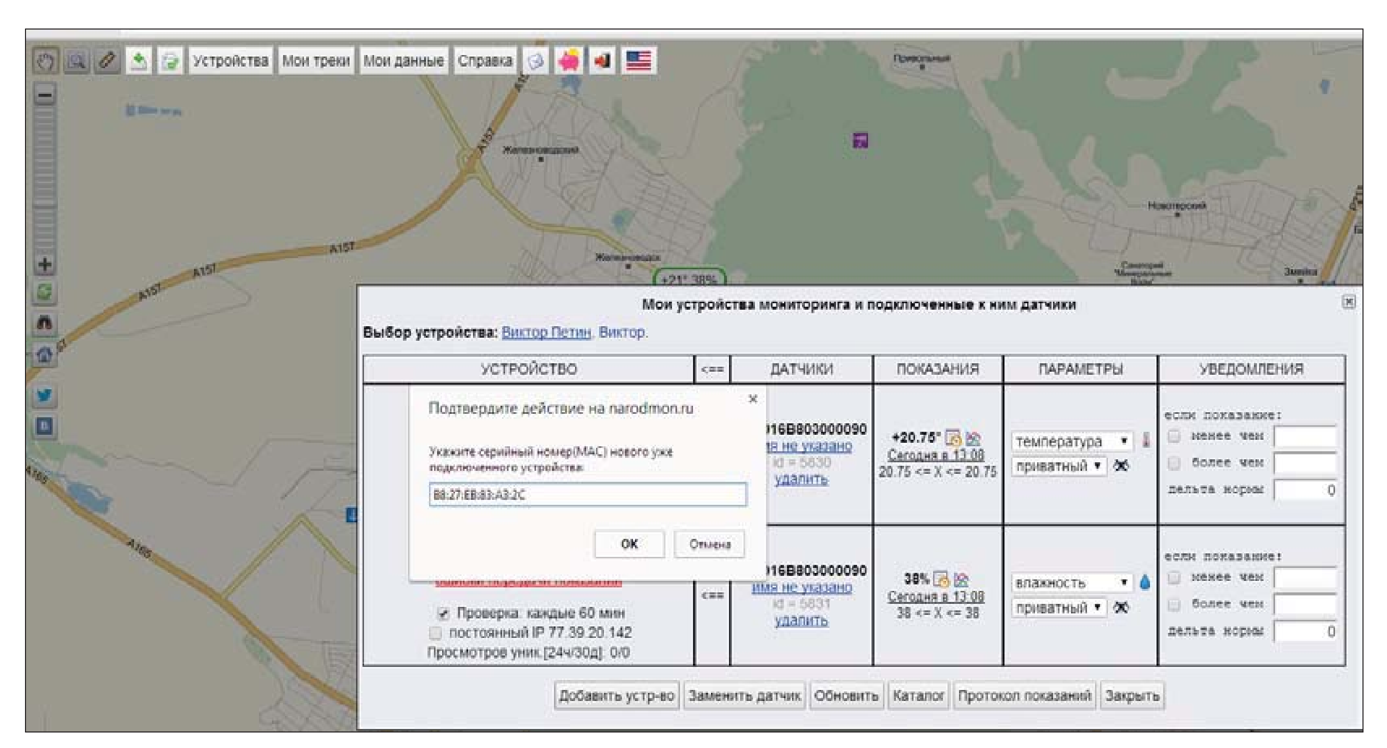


Рис. 4.48. Добавление датчиков на сайте **narodmon.ru**

Скрипт cron_narodmon.php мы будем запускать по командам cron каждые 5 минут. Открываем файл конфигурации cron:

```
crontab -e
```

И записываем в него строку:

```
*/5 * * * * /usr/bin/php -q /home/pi/cron_narodmon.php > /dev/null 2>&1
```

Теперь наши датчики расположены на карте Народного мониторинга (рис. 4.49), и каждые 5 минут отправляют данные на сайт.

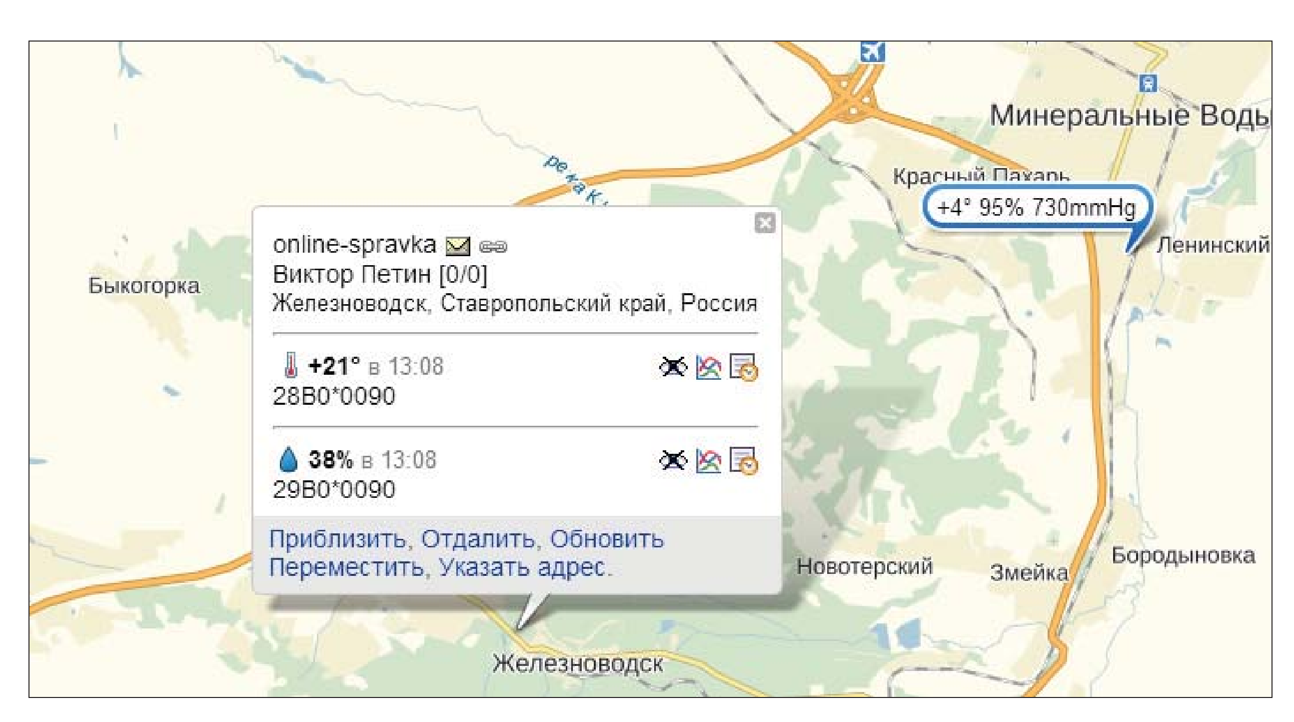


Рис. 4.49. Наши датчики на карте **narodmon.ru**

> ***Примечание***
>
> Файл скрипта *cron_narodmon.php* а также другие вспомогательные файлы этого примера находятся в папке *glava_04\narodmon* сопровождающего книгу электронного архива (см. *приложение*).

4.17. Размещение изображения с камеры Raspberry Pi на сайте Народного мониторинга

Для размещения изображения с камеры на сайте Народного мониторинга настроим трансляцию с веб-камеры в сеть с помощью пакета MJPG-streamer (см. *разд. 4.11*). Изменим порт для отдачи потокового видео на 8088:

```
cd mjpg-streamer
sudo nano mjpg-streamer.sh
```

И установим:

```
PORT="8080"
```

Для запуска трансляции запускаем bash-скрипт mjpg-streamer.sh:

```
sudo ./mjpg-streamer.sh start
```

Видеопоток станет доступен по URL следующего вида:

```
http://raspberrypi:8088?action=stream
```

Скриншот доступен по URL следующего вида:

```
http://raspberrypi:8088?action=snapshot
```

Поскольку доступ к камере необходим из внешней сети, следует открыть на роутере порт 8088 (рис. 4.50).

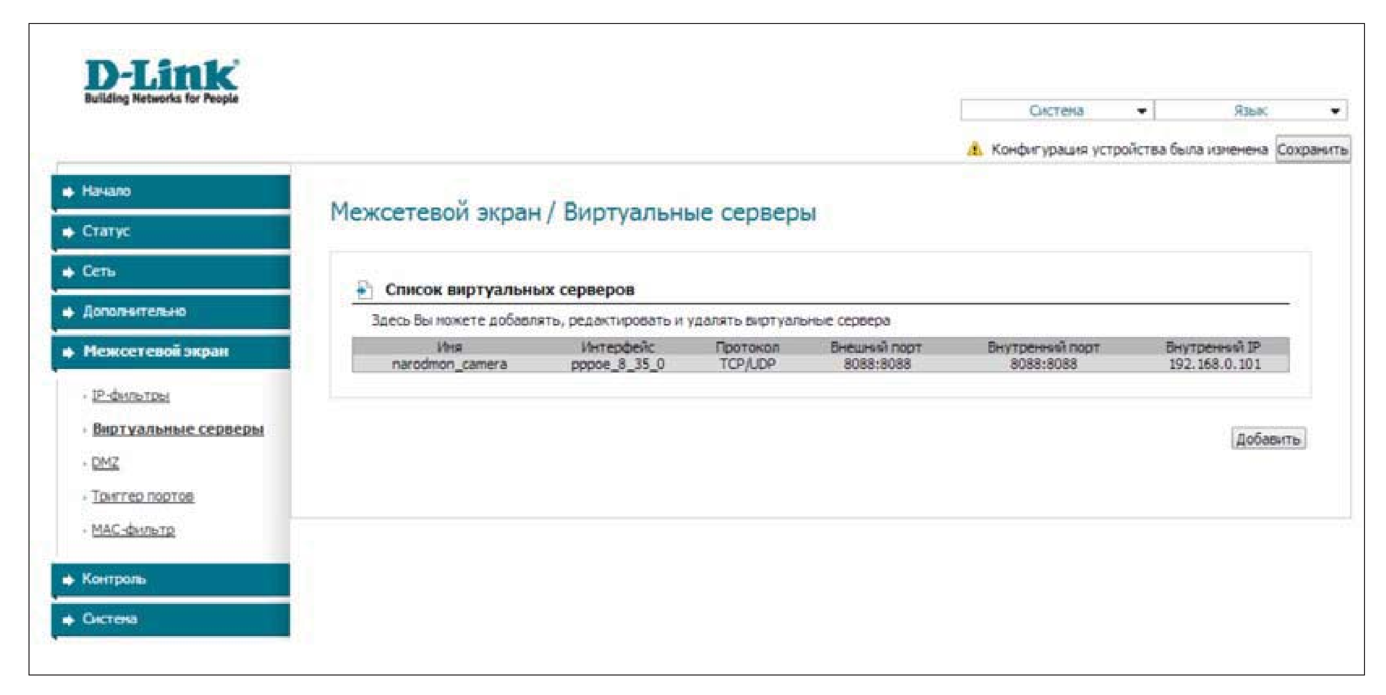


Рис. 4.50. Открытие порта 8088 на роутере DSL-2500U

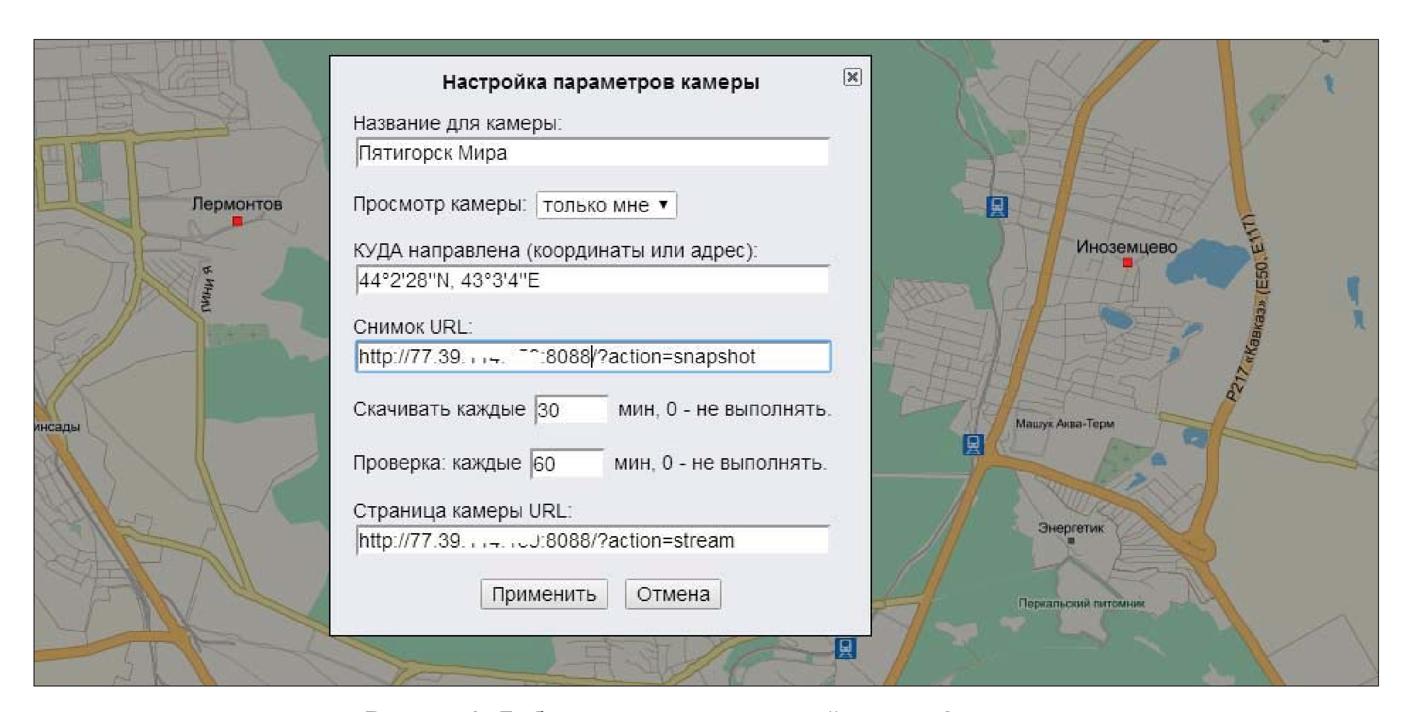


Рис. 4.51. Добавление камеры на сайте **narodmon.ru**

Теперь заходим в свой профиль на сайте Народного мониторинга (http://www.narodmon.ru) и выбираем пункт меню **Устройства | Мои камеры | Добавить камеру**. Вводим необходимые данные (рис. 4.51). Если все настроено правильно, камера будет добавлена на карту.

Поскольку в настройках камеры установлен только приватный просмотр, камера пока видна лишь из моего профиля. Для настройки публичного доступа необходимо в меню **Просмотр камеры** выбрать пункт **всем**.



ГЛАВА 5

Медиапроигрыватель Xbox Media Center (XBMC)

Благодаря своей дешевизне, низкому энергопотреблению и небольшому размеру Raspberry Pi может стать основой для мощного медиацентра, построенного на основе XBMC (**Xb**ox **M**edia **C**enter).

XBMC — это открытый проект по созданию медиацентра, изначально предназначенного для игровой приставки Xbox. Однако в настоящее время проектом поддерживается множество платформ, в том числе Android, Windows и Linux. Поскольку XBMC представляет собой проект с открытым исходным кодом, сообщество поклонников мини-ПК Raspberry Pi предлагает для него готовые сборки пакетов XBMC.

С помощью XBMC Media Center можно работать с музыкальными и видеофайлами, просматривать HD-видео, RSS, фотографии и фильмы. XBMC Media Center понимает файлы форматов MPEG-1/2/4, XviD, DivX, MP3, JPG, AAC, GIF и многие другие. Кроме того, в функциональные возможности XBMC Media Center входит воспроизведение CD и DVD, которые он может читать как файлы, захваченные с жесткого диска.

5.1. Установка дистрибутива Raspbmc

Медиацентр, основанный на XBMC, включает сразу несколько дистрибутивов операционных систем на базе Linux. Самыми популярными из них являются Raspbmc и OpenELEC, разработанные специально для Raspberry Pi.

OpenELEC представляет собой урезанную и оптимизированную операционную систему, предназначенную специально для запуска Xbox Media Center. OpenELEC не содержит ничего, кроме необходимого минимума для выполнения основных ее функций. Поэтому достаточно трудно, например, установить на нее другие пакеты. Дистрибутив Raspbmc имеет гораздо больше возможностей для работы с операционной системой, его мы здесь и рассмотрим.

Дистрибутив Raspbmc укомплектован медиацентром XBMC 12.0 и предлагает "из коробки" полный спектр средств для создания домашнего кинотеатра, поддержи-

вающего работу с видео HD-качества (1080p). При воспроизведении видео задействуются средства аппаратного декодирования h.264, а также — на основе отдельно приобретаемых лицензий — MPEG-2 и VC-1, предоставляемые графическим ускорителем Broadcom VideoCore.

Дистрибутив имеет встроенную поддержку различных типов систем удаленного управления, в том числе и инфракрасного приемника, подключаемого через порт GPIO мини-ПК Raspberry Pi. Дополнительно поддерживается организация управления XBMC по сети с мобильного телефона при помощи специализированных приложений для платформ iOS и Android. Для обеспечения трансляции музыки и видео с устройств Apple дистрибутивом поддерживаются технологии AirPlay и AirTunes.

Сетевое соединение может быть организовано как по проводной, так и по беспроводной сети. В состав дистрибутива интегрированы сервисы для организации доступа к локальным коллекциям контента по протоколам Samba, NFS, TVHeadend, FTP, HTTP и SSH. По умолчанию правила межсетевого экрана настроены на возможность доступа только из локальной сети. Поддержанию системы в актуальном состоянии способствует система автоматической установки обновлений.

Как мы помним из *главы 2*, инструмент NOOBS в числе прочих предоставляет возможность установки и дистрибутива Raspbmc. Так что в меню выбора операционной системы выбираем **RaspBMC** и ждем окончания установки.

В главном окне установленной и загруженной системы Raspbmc (рис. 5.1) имеется пять вкладок: **Погода**, **Фото**, **Видео**, **Музыка**, **Программы** и **Система**.



Рис. 5.1. Главное окно Raspbmc

5.2. Установка начальных параметров

Для установки языка и региональных параметров выполним команду **Система | Настройки | Внешний вид | Языковые настройки**. Выберем: **Язык — Russian**, **Регион — Russia**, **Кодировка — Cyrillic (ISO)**, **Страна в часовом поясе — Russia** и свой **Часовой пояс** (рис. 5.2).

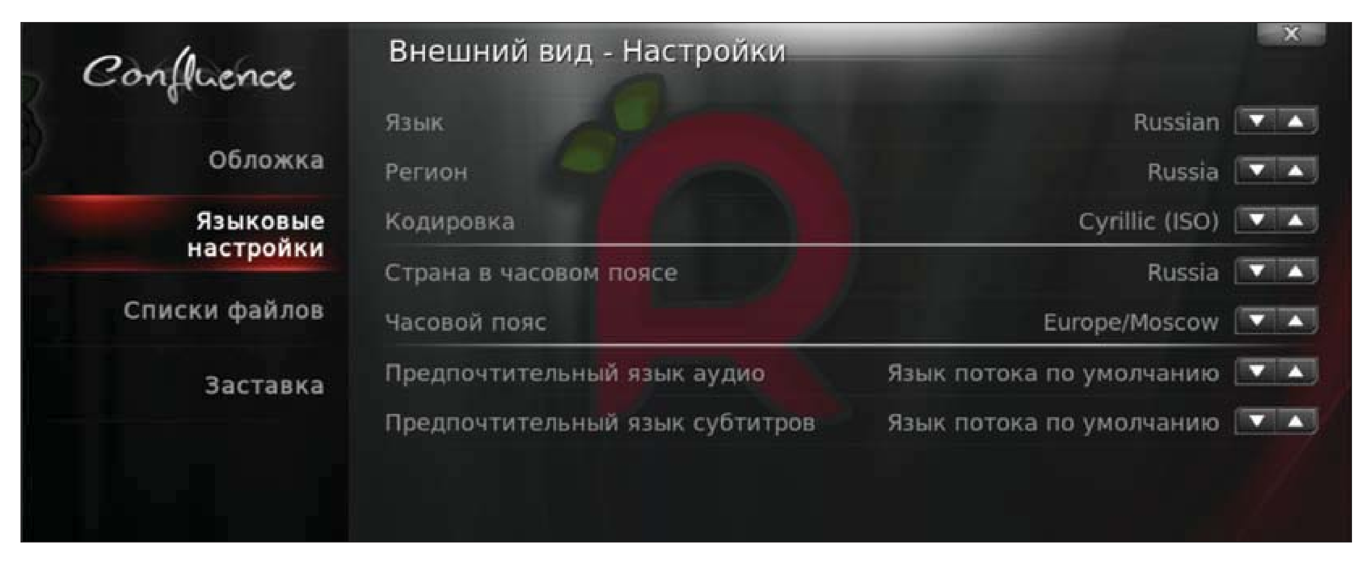


Рис. 5.2. Выбор языковых и региональных настроек

В процессе работы над книгой мне приходилось делать много скриншотов. Для снятия скриншота в Raspbmc надо нажать клавишу <PrintScreen>. Поскольку скриншоты удобно хранить все вместе, выполним команду **Система | Настройки | Отладка** и создадим для скриншотов отдельную папку: **/home/pi/screenshots** — после чего зайдем в нее и нажмем кнопку **OK** (рис. 5.3).

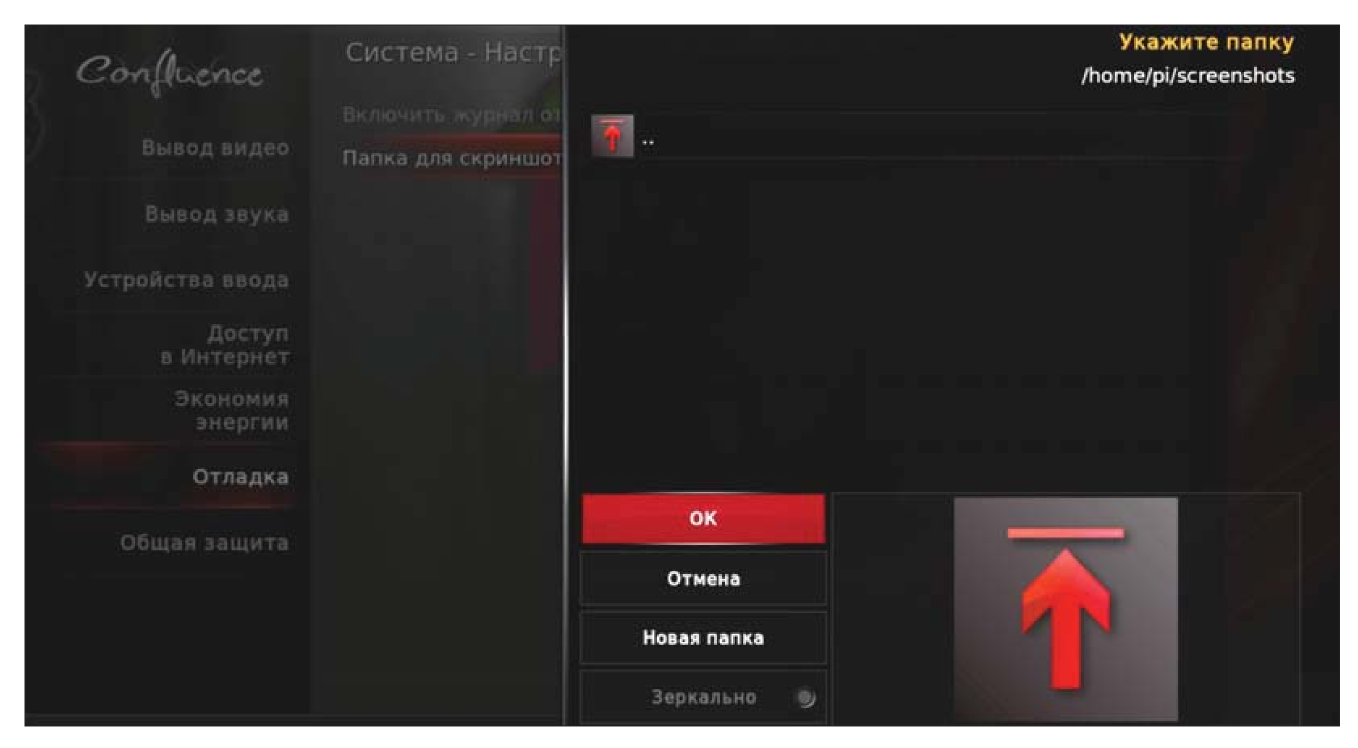


Рис. 5.3. Назначение папки для сохранения скриншотов

Для использования всех возможностей медиаплеера Raspbmc требуется подключение к Интернету. Если необходимо, настраиваем параметры прокси-сервера для соединения с Интернетом через меню: **Система | Настройки | Доступ в Интернет** (рис. 5.4).

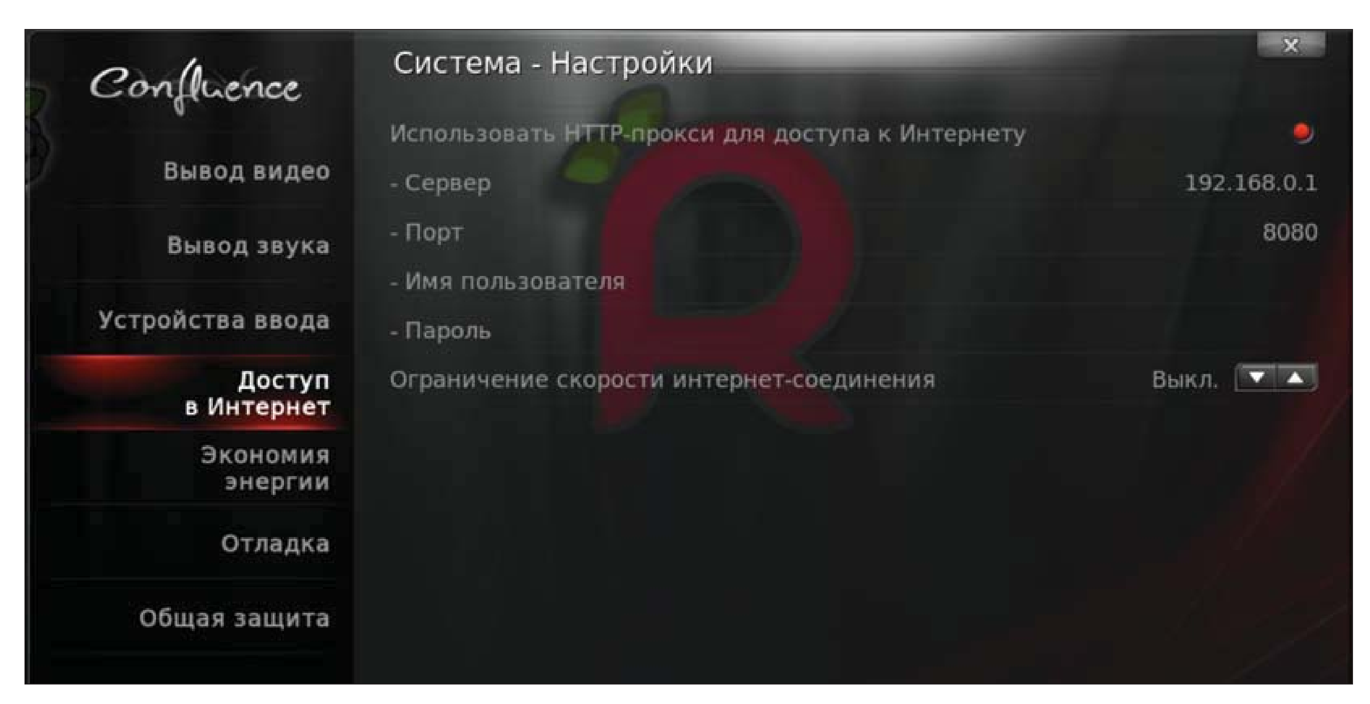


Рис. 5.4. Настройка прокси-сервера

Непосредственно сами сетевые настройки можно задать на вкладке **Network Configuration** программы **Raspbmc settings**, запускаемой из меню **Система | Настройки | Программы** (рис. 5.5).

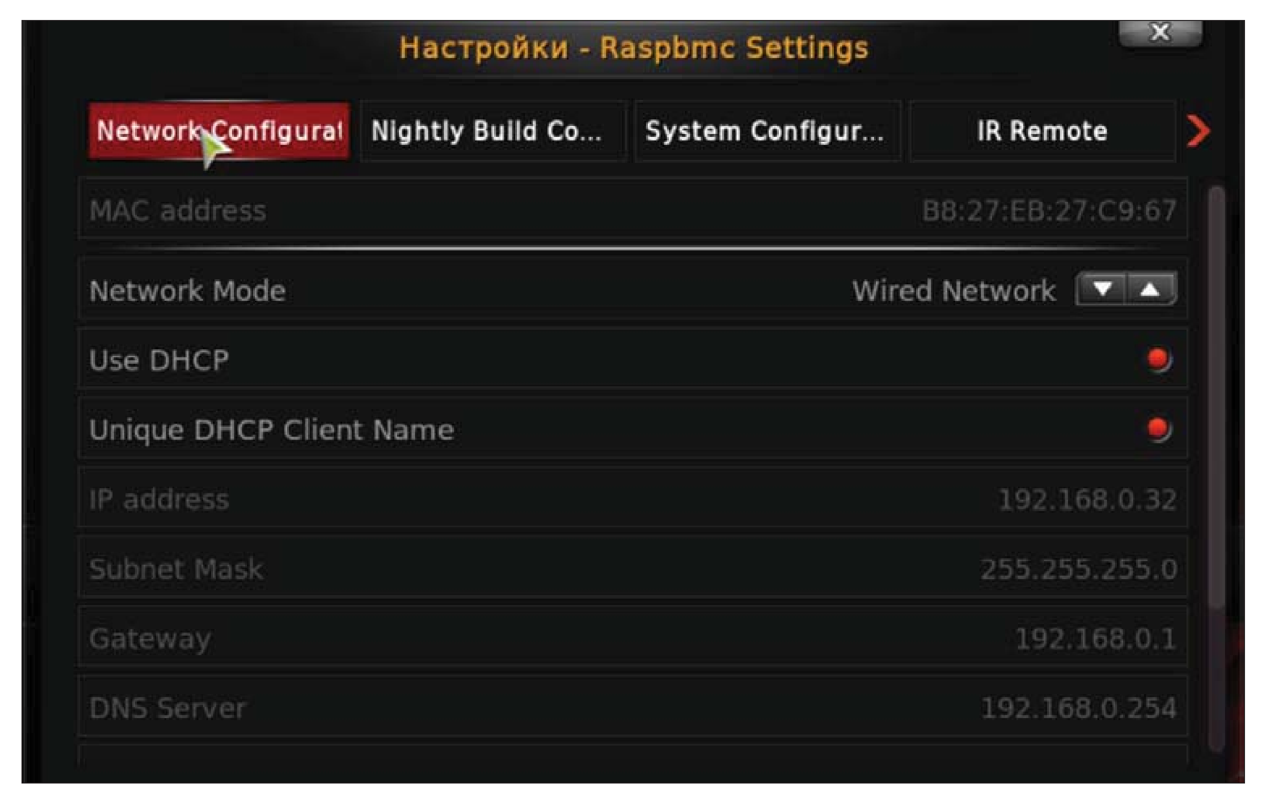


Рис. 5.5. Настройка параметров сети

Здесь же, на вкладке **System Configuration**, выбираем опции для использования системы Raspbmc в качестве FTP-сервера, SSH-сервера и сервера Samba (рис. 5.6). Поскольку к SSH мы будем обращаться для настройки Raspbmc удаленно, необходимо задать для доступа к серверу SSH логин и пароль:

- login — `pi`;
- password — `raspberry`.

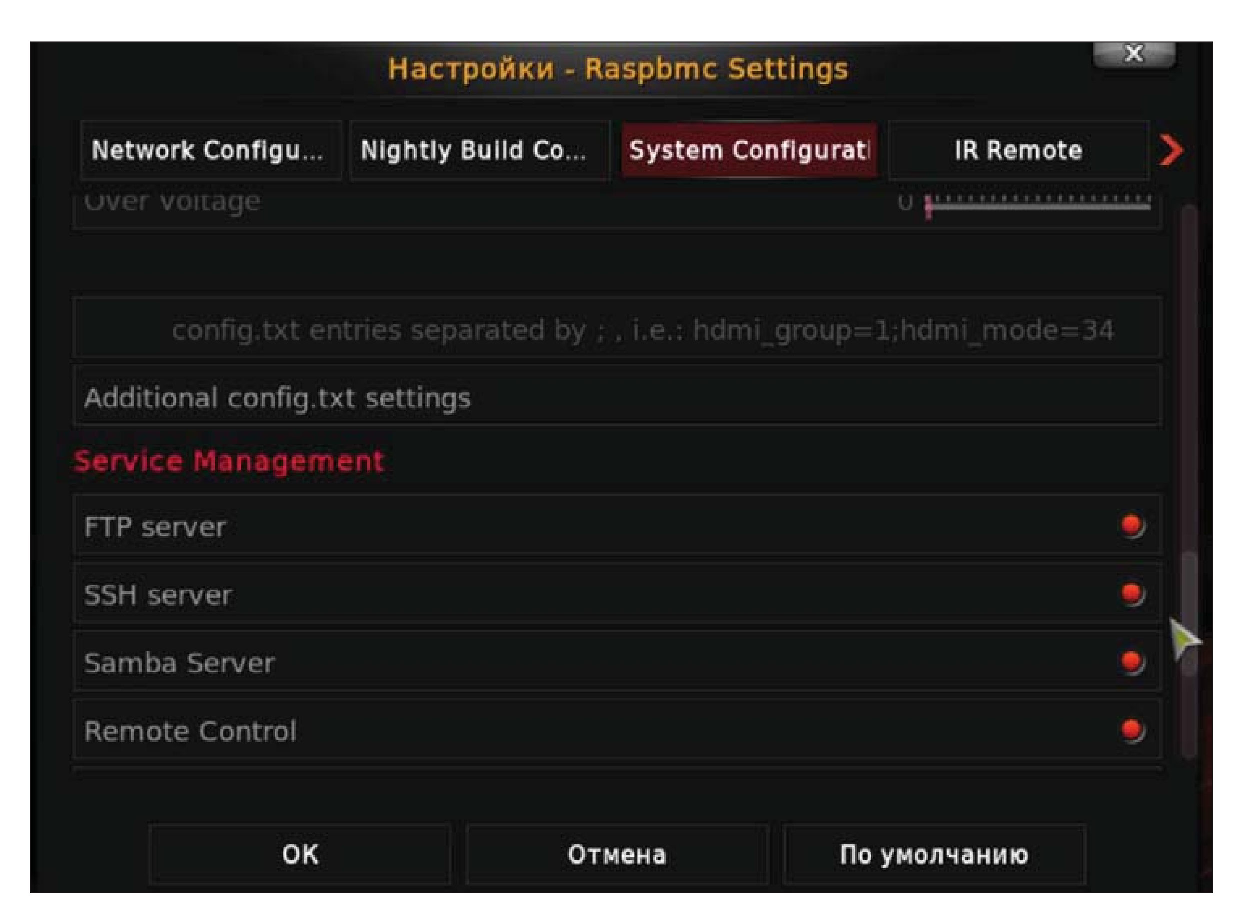


Рис. 5.6. Настройка **System Configuration**

5.3. Новостная лента

В нижней части окна Raspbmc расположена бегущая строка RSS-ленты. Сюда можно добавить свои любимые сайты для получения с них свежей информации. В главном окне системы заходим в меню **Программы | Еще**, из списка выбираем **RSS Editor** (рис. 5.7) и в открывшемся меню — пункт **Установить** (рис. 5.8).

Установив RSS Editor, идем в меню **Система | Настройки | Внешний вид | Обложка**) и выбираем пункты **Включить RSS новости** и **Изменить**. И попадаем в окно списка новостных лент (рис. 5.9).

В отобразившемся списке установленных RSS-лент нажимаем кнопку **Добавить** и вводим данные RSS-канала (рис. 5.10).

Затем выбираем интервал обновления новостей (рис. 5.11).

И получаем ленту новостей на главной странице (рис. 5.12).

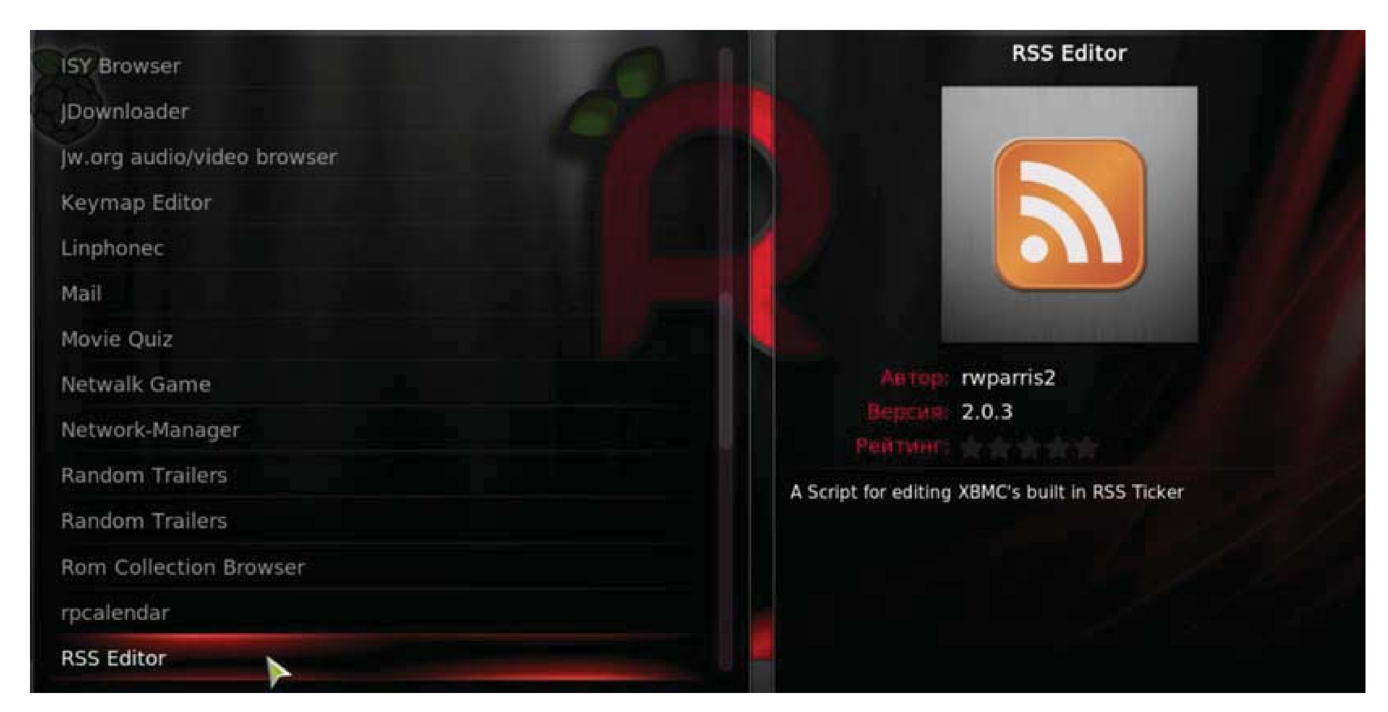


Рис. 5.7. Список программ

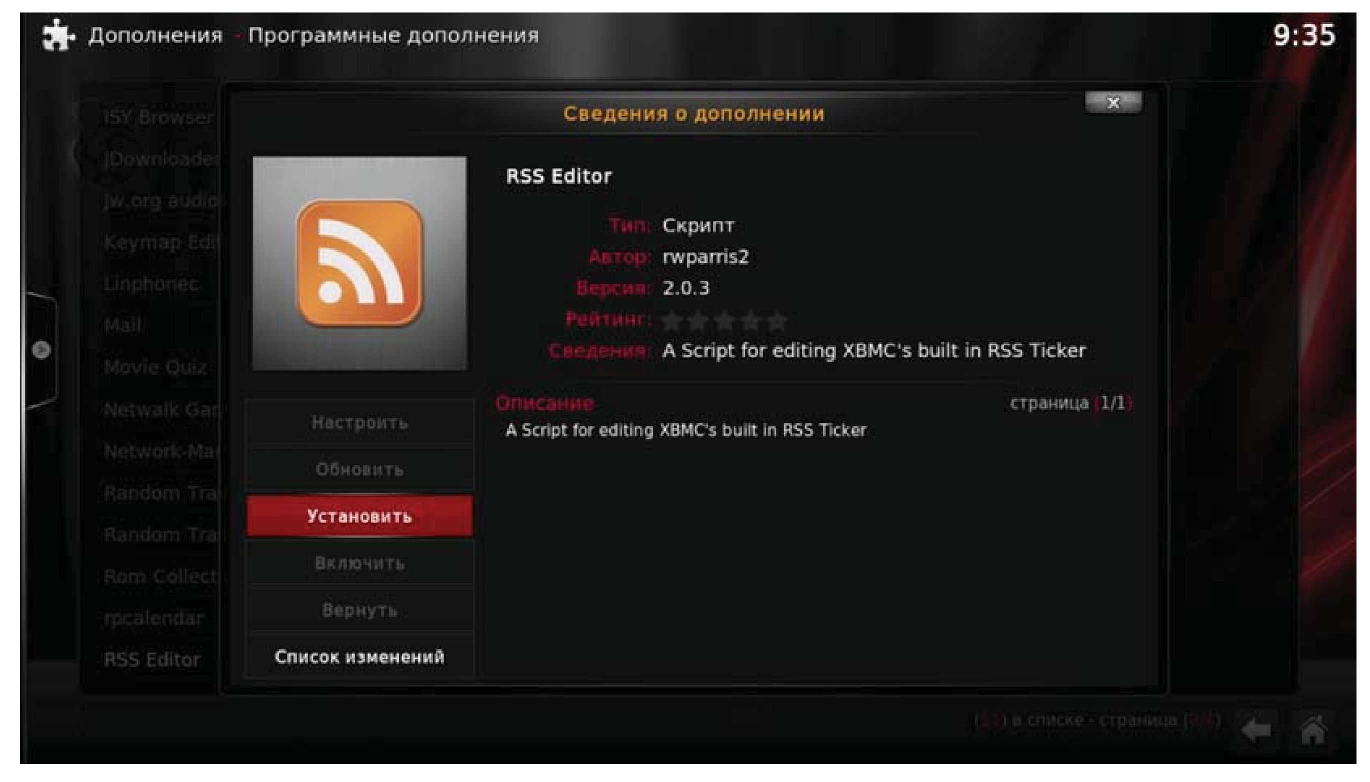


Рис. 5.8. Установка RSS Editor



Рис. 5.9. Список установленных RSS-лент



Рис. 5.10. Добавление RSS-канала

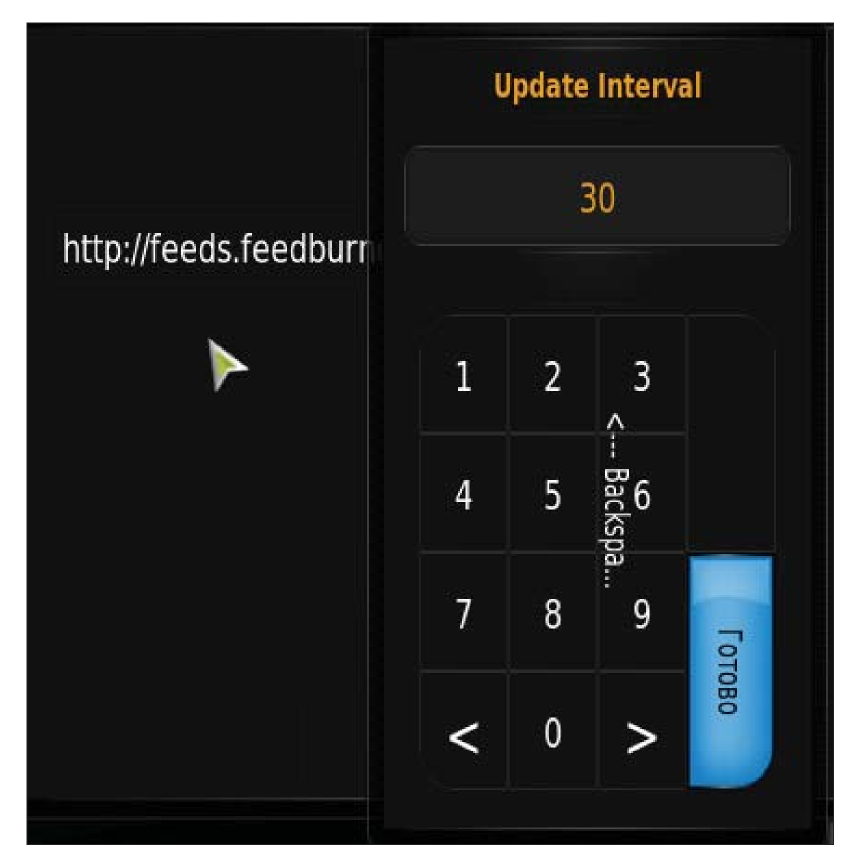


Рис. 5.11. Установка интервала обновления RSS-канала

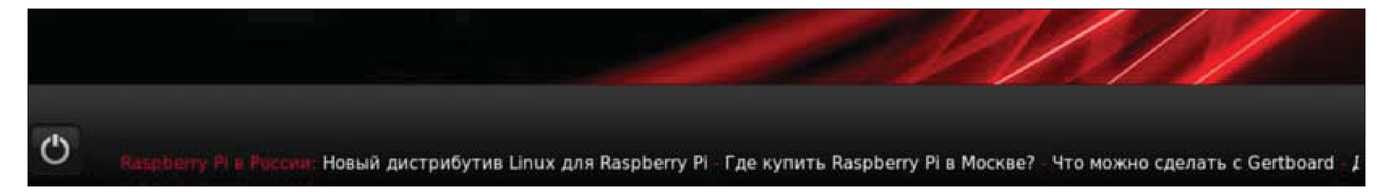


Рис. 5.12. Отображение RSS-новостей на главной странице Raspbmc

5.4. Погода

Система Raspbmc позволяет получать прогноз погоды для трех городов из каждого источника данных о погоде. Для настройки отображения такого прогноза перейдем по меню **Система | Настройки | Погода** (рис. 5.13). По умолчанию здесь уже заранее назначен источник **Weather Underground** (прогноз погоды с сайта **wunderground.com**).

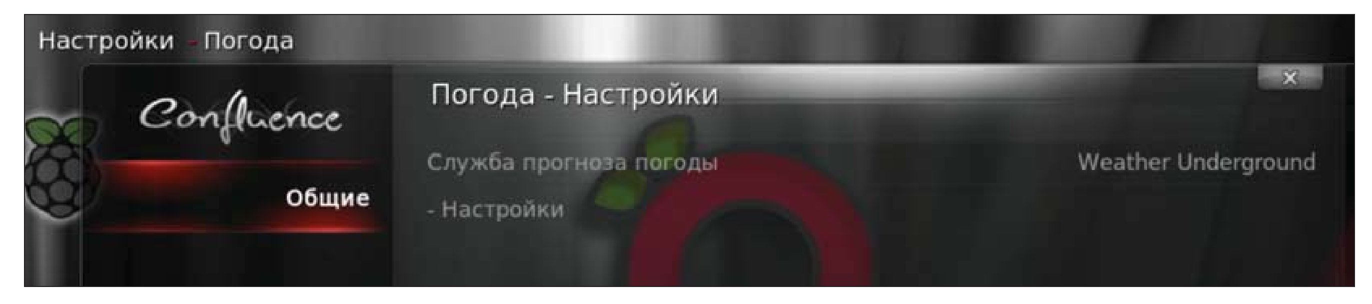


Рис. 5.13. Настройка погоды



Рис. 5.14. Список источников прогноза погоды

Выбрав в этом окне пункт **Служба прогноза погоды**, можно перейти на выбор других возможных для добавления источников погоды (рис. 5.14).

Затем, выбрав в меню настройки погоды (см. рис. 5.13) пункт **Настройки**, мы перейдем в окно выбора городов, для которых будет показываться прогноз (рис. 5.15). Здесь для каждого выбранного источника прогноза погоды можно назначить несколько городов.

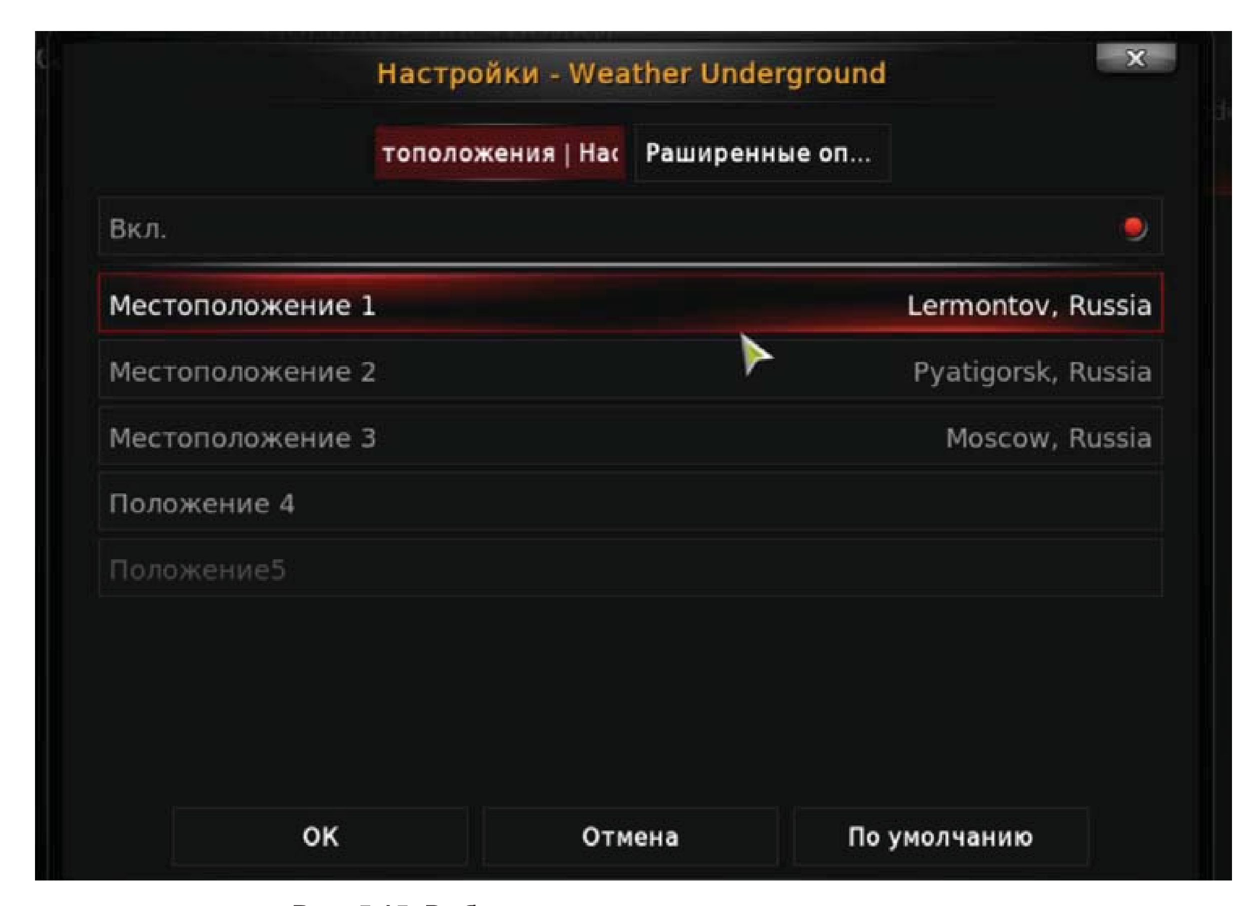


Рис. 5.15. Выбор городов для показа прогноза погоды

Теперь, при выборе в главном окне системы пункта меню **Погода**, мы можем увидеть прогноз погоды для первого из предусмотренных городов (рис. 15.16). Прогноз состоит из двух вкладок: на одной — текущая погода, на другой — прогноз на 10 дней.



Рис. 5.16. Окно отображения погоды

На выезжающей из левой границы окна вкладке можно выбрать следующий город и уточнить настройки для прогноза (рис. 5.17).



Рис. 5.17. Вкладка для настройки параметров отображения погоды

5.5. Подключение репозитория русскоязычных дополнений

Рассматриваемая нами оболочка XBMC имеет удобную систему дополнений и специальный менеджер для них, что позволяет легко находить и устанавливать дополнения прямо из интерфейса программы. По умолчанию подключен официальный источник под названием **XBMC.org Add-ons**, но можно воспользоваться и любым сторонним хранилищем, список которых расположен на странице: **http://wiki.xbmc.org/index.php?title=Unofficial_Add-on_Repositories**.

Советую обратить внимание на репозиторий под названием **seppius-xbmc-repo**, который добавляет в программу поддержку популярных русскоязычных медиаресурсов. Для подключения репозитория необходимо скачать его файл repository.seppius.zip. Итак, подключаемся к нашему мини-ПК Raspberry Pi на ОС Raspbmc по SSH (напоминаю: логин — `pi`, пароль — `raspberry`) и скачиваем ZIP-файл репозитория по ссылке: **http://seppius-xbmc-repo.googlecode.com/files/repository.seppius.zip** (рис. 5.18).

```
pi@raspbmc: ~
Файл Правка Вид Поиск Терминал Справка
petin@ubuntu1:~$ ssh -X pi@192.168.0.32
pi@192.168.0.32's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
sh: 1: /usr/bin/xauth: not found
pi@raspbmc:~$ wget http://seppius-xbmc-repo.googlecode.com/files/repository.seppius.zip
--2013-11-12 13:30:24--  http://seppius-xbmc-repo.googlecode.com/files/repository.seppius.zip
Resolving seppius-xbmc-repo.googlecode.com (seppius-xbmc-repo.googlecode.com)... 74.125.143.82, 2a00:1450:4010:c0
4::52
Connecting to seppius-xbmc-repo.googlecode.com (seppius-xbmc-repo.googlecode.com)|74.125.143.82|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 239075 (233K) [application/x-zip]
Saving to: `repository.seppius.zip'

100%[======================================================================>] 239,075      233K/s   in 1.0s

2013-11-12 13:30:26 (233 KB/s) - `repository.seppius.zip' saved [239075/239075]

pi@raspbmc:~$
```

Рис. 5.18. Скачивание ZIP-файла репозитория **seppius-xbmc-repo**

Теперь в окне Raspbmc переходим по меню **Система | Дополнения | Установить из файла ZIP** и указываем скачанный файл (рис. 5.19).

После подключения репозитория **seppius-xbmc-repo** в списке дополнений появится много русскоязычных медиаресурсов.

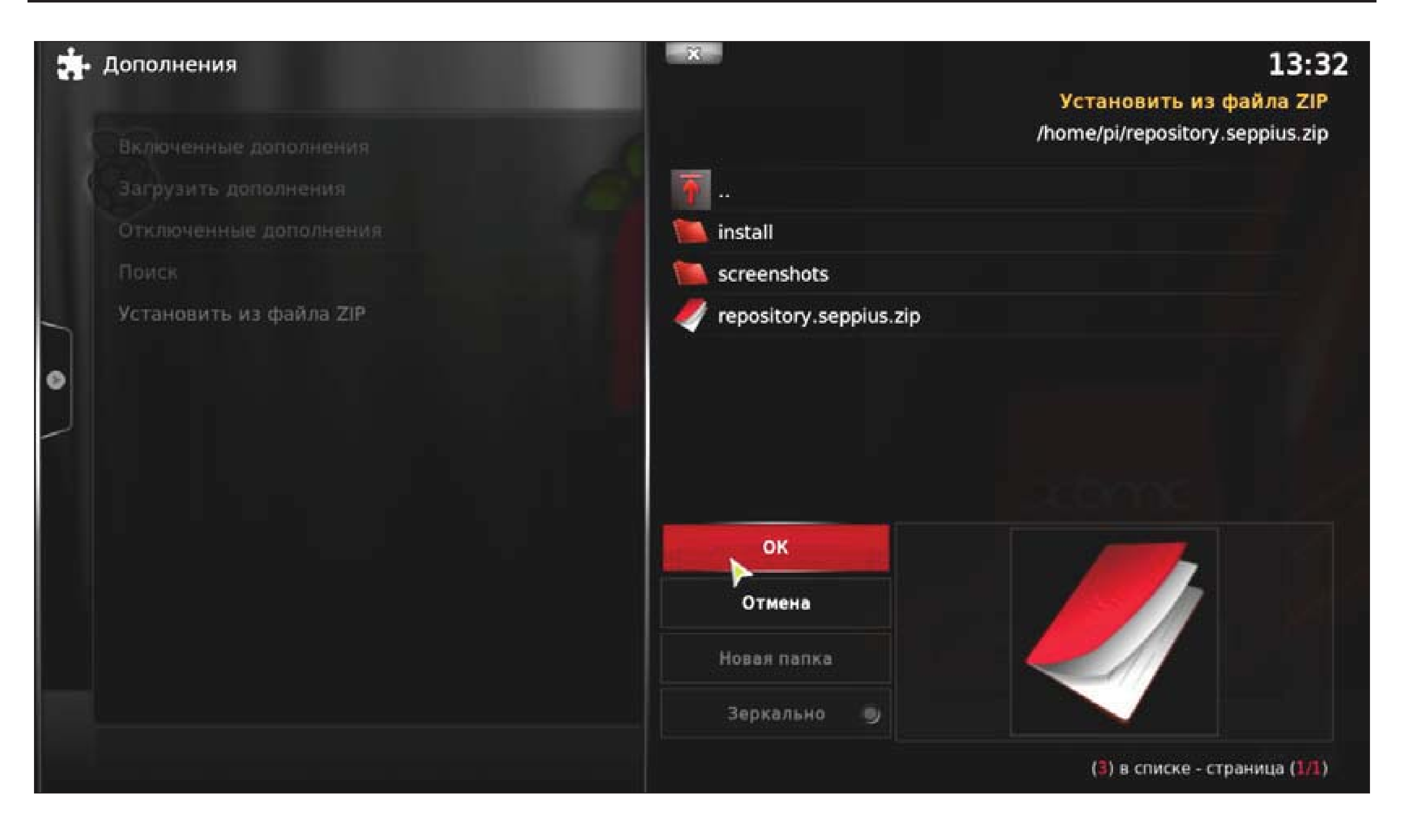


Рис. 5.19. Подключение репозитория **seppius-xbmc-repo**

5.6. Фото

Меню **Фото** главного окна Raspbmc позволяет просматривать изображения форматов RAW, BMP, JPEG, GIF, PNG, TIFF, MNG, ICO, PCX и Targa/TGA. Для добавления источника просмотра фотографий надо выполнить команду меню **Фото | Добавить источник** (рис. 5.20). Выбираем путь к источнику показа фотографий и даем ему название (рис. 5.21).

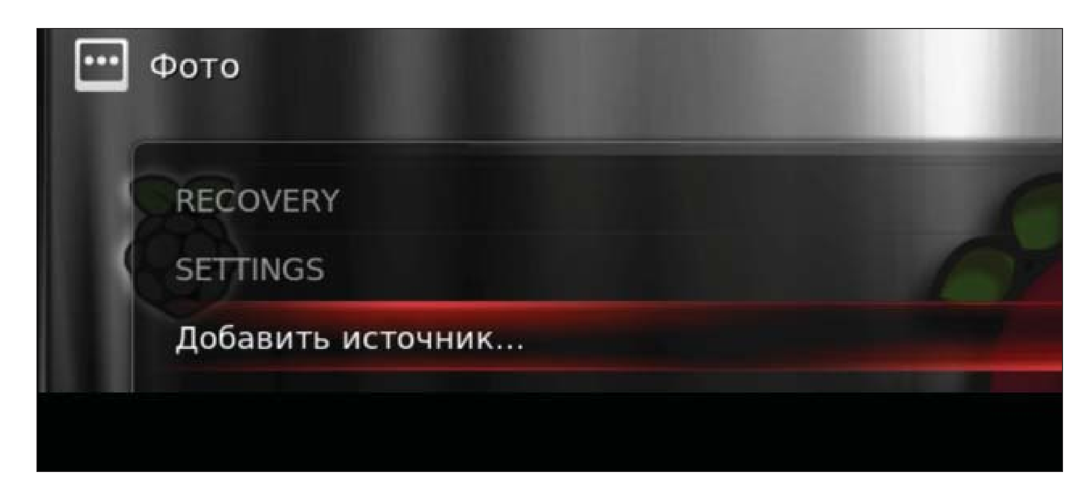


Рис. 5.20. Добавление источника просмотра фотографий

Созданный нами источник появляется в списке (рис. 5.22). Настройки показа фотографий осуществляются в меню **Система | Настройки | Фото** (рис. 5.23).

При выборе источника фотографий загружается список содержащихся в нем фотографий с показом миниатюры (превью) выбранной (рис. 5.24).

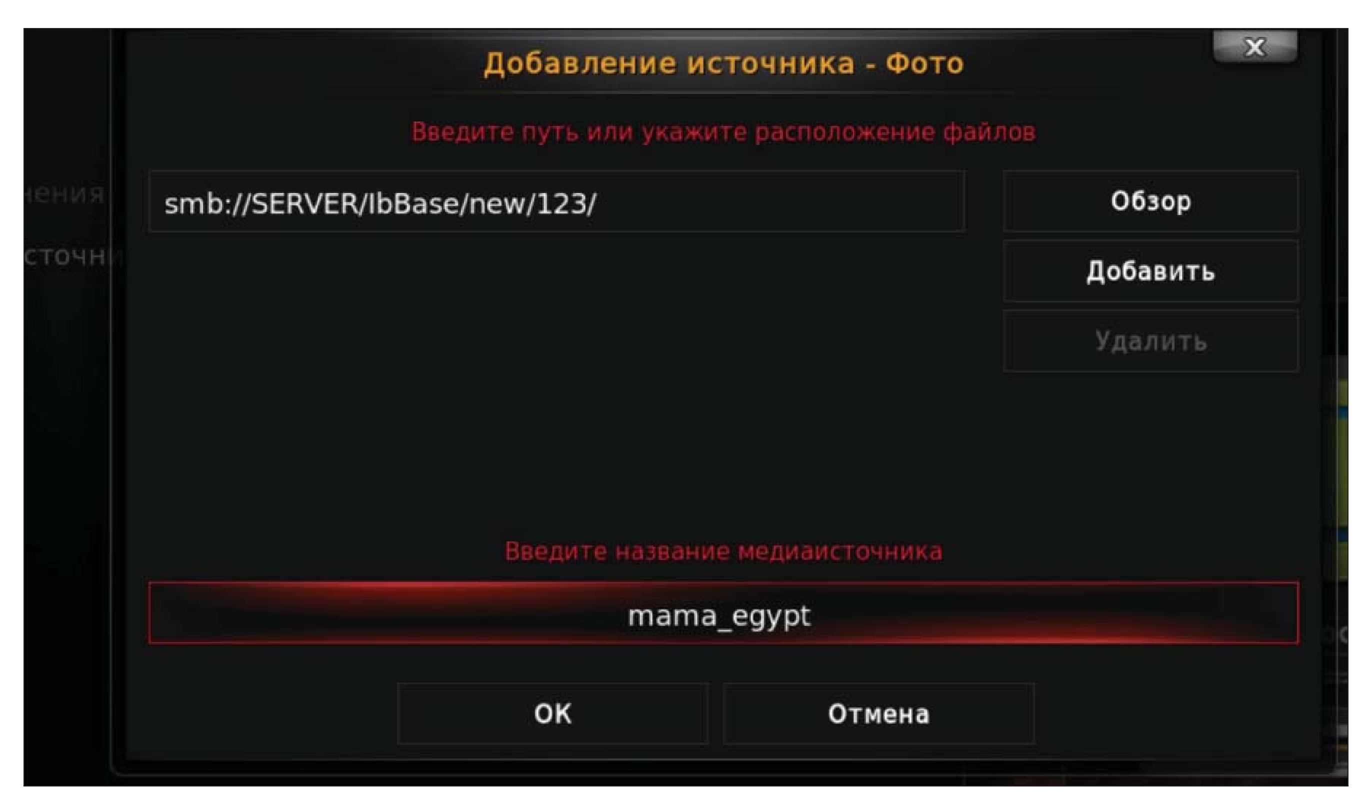


Рис. 5.21. Путь к источнику фотографий



Рис. 5.22. Источник фотографий добавлен

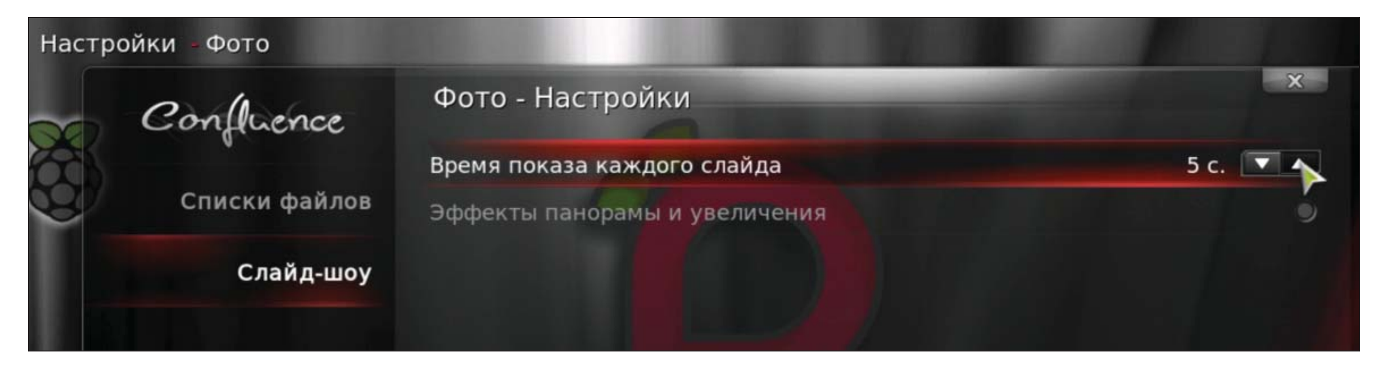


Рис. 5.23. Настройки показа фото

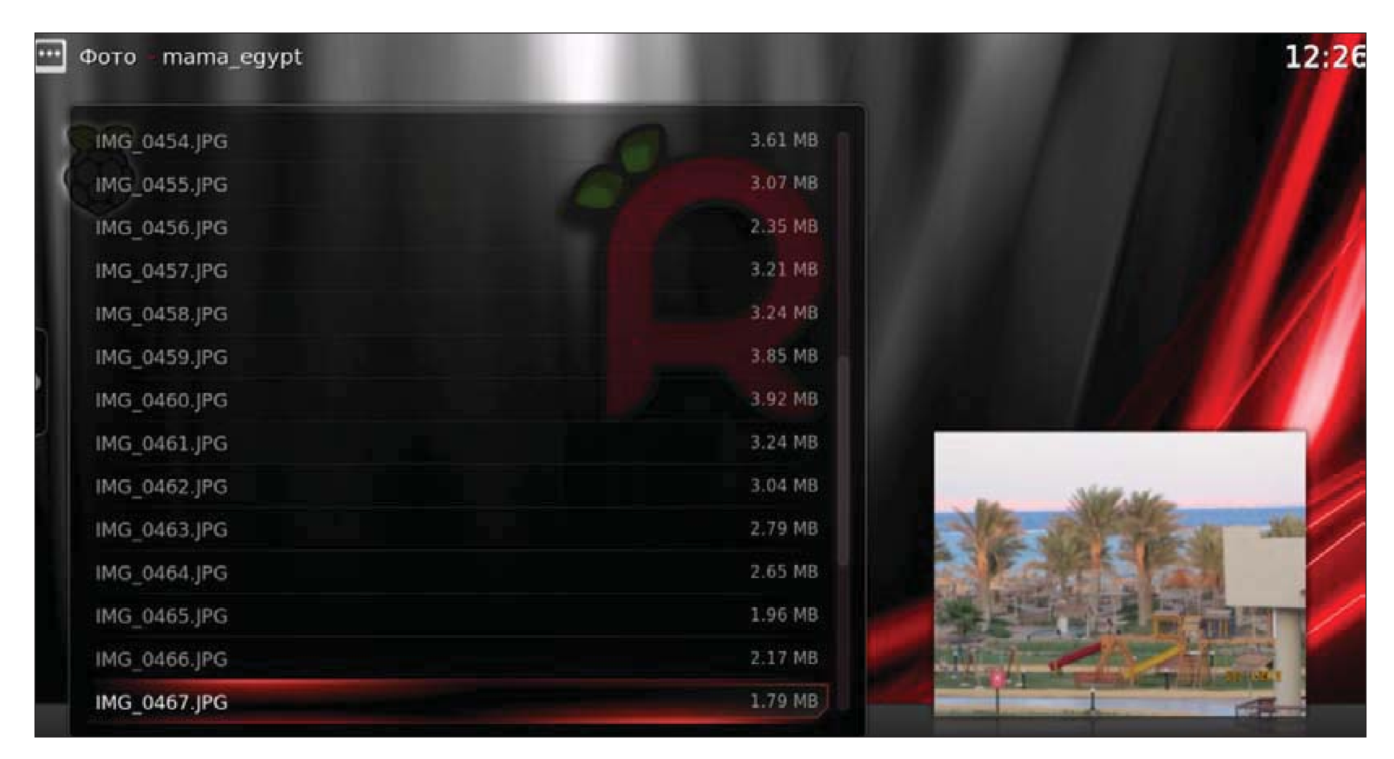


Рис. 5.24. Выбор источника фотографий для просмотра

Запустить просмотр фотографий можно и в режиме слайд-шоу. Настройки просмотра для выбранного источника выставляются в выезжающем из левой границы окна меню (рис. 5.25).

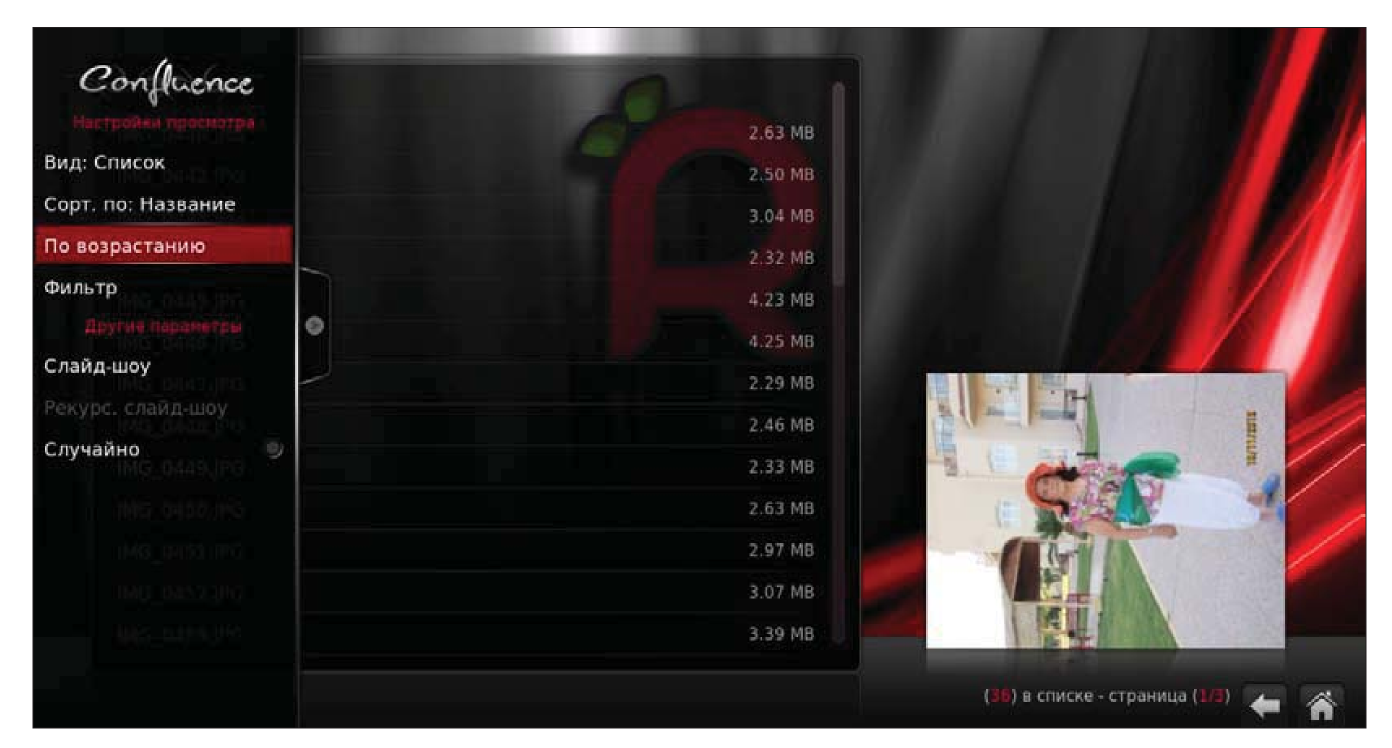


Рис. 5.25. Выбор источника фотографий для просмотра в режиме слайд-шоу

Среди интересных дополнений раздела **Фото** — плагин **google**, позволяющий искать, просматривать и сохранять изображения средствами Google Image Search. Для установки этого дополнения проходим по меню **Фото | Фотодополнения | Еще** и выбираем плагин **google** (рис. 5.26).



Рис. 5.26. Добавление плагина **google**

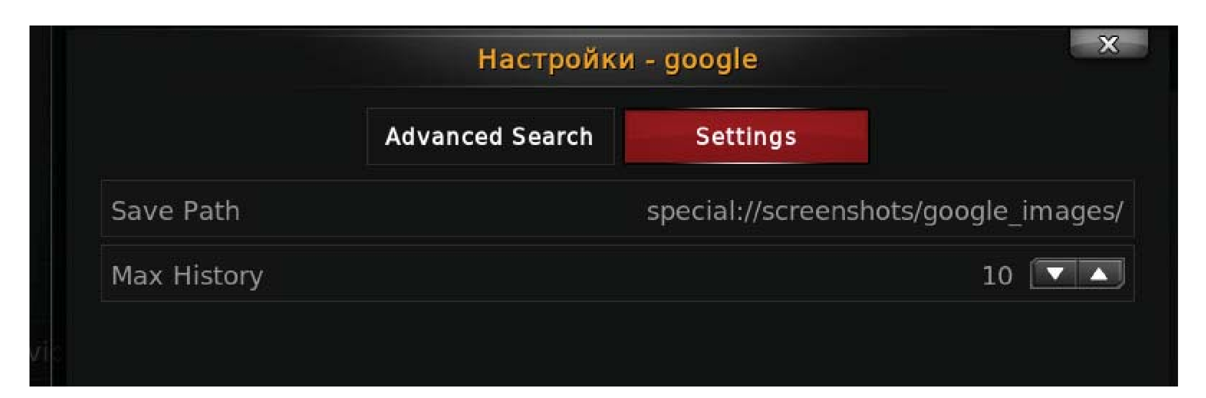


Рис. 5.27. Настройки плагина **google**

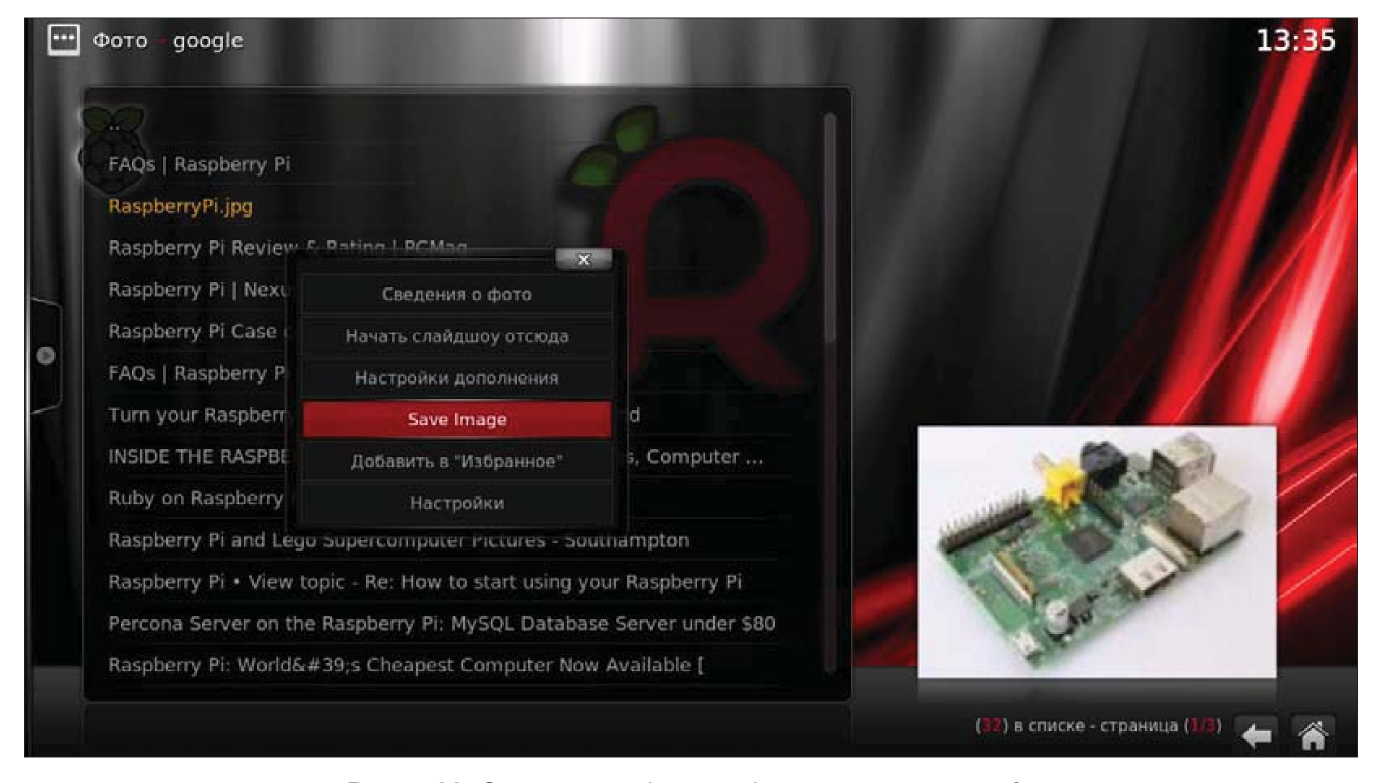


Рис. 5.28. Сохранение фотографии из плагина **google**

Загрузив плагин, выбираем пункт меню **Настройки** и указываем путь для сохранения фотографий (рис. 5.27).

Теперь по выбранному пути можно сохранять найденные с помощью плагина **google** изображения (рис. 5.28).

Имеются в разделе **Фото** и другие интересные дополнения, в том числе поддержка онлайновых сервисов Flickr и Picasa. Мне также понравилось расширение **The Big Picture**, которое предоставляет доступ к фоторепортажам с сайтов **Boston.com**, **NewYorkTimes.com** и др. (рис. 5.29)

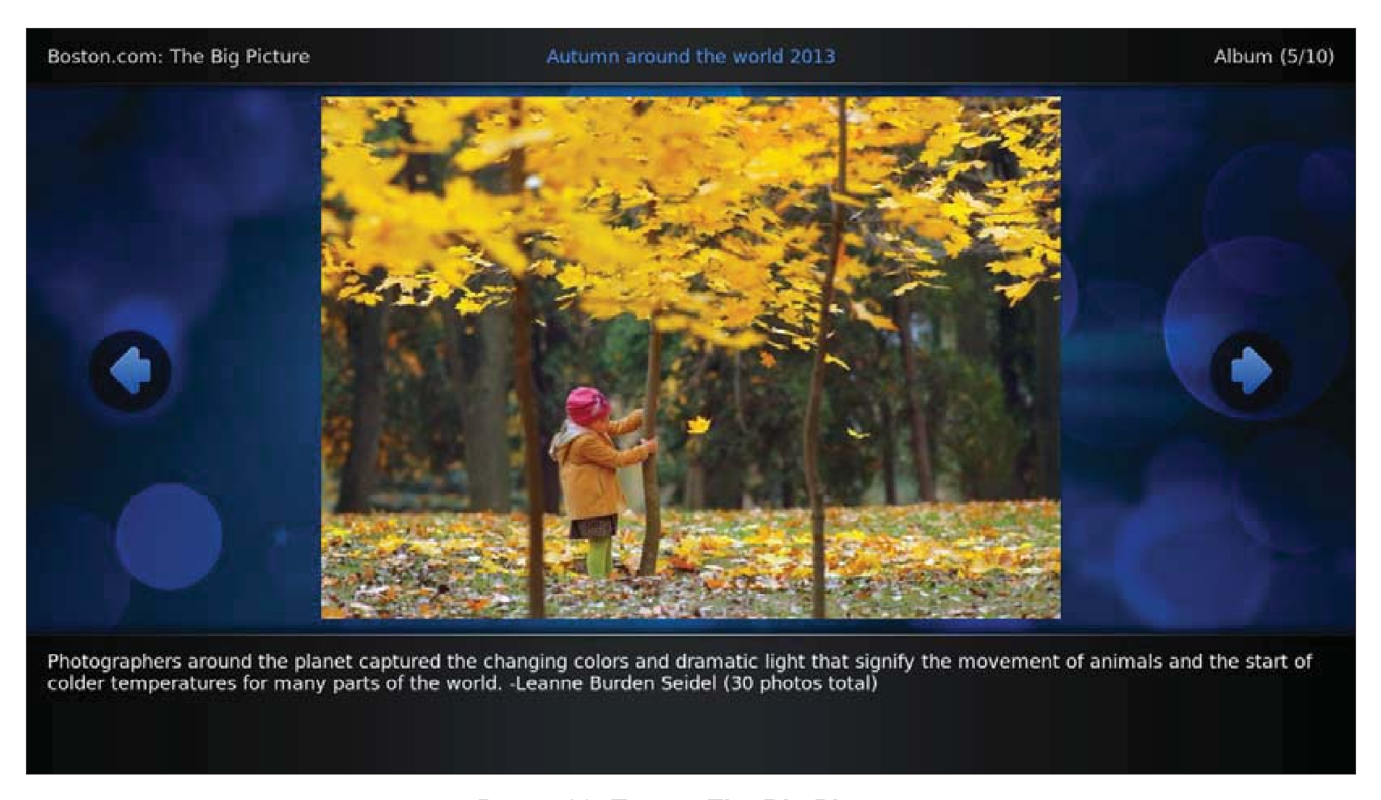


Рис. 5.29. Плагин **The Big Picture**

5.7. Музыка

XBMC поддерживает воспроизведение аудио следующих форматов: MIDI, AIFF, WAV/WAVE, MP2, MP3, AAC, AACplus, AC3, DTS, ALAC, AMR, FLAC, Monkey's аудио (APE), RealAudio, SHN, WavPack, MPC/Musepack/MPEG +, Speex, Vorbis и WMA.

Подключать Raspberry Pi к телевизору лучше всего с помощью HDMI-кабеля, потому что в этом случае звук и видео будут передаваться по одному кабелю. Впрочем, можно подключить звук отдельно через выход для наушников. Параметры звука определяются в меню **Система | Настройки | Вывод звука** (рис. 5.30). Здесь можно определить, какой выход использовать для звука — HDMI или аналоговый, а также выбрать формат передачи звука — AC3 или DTS.

В качестве источника музыки с помощью меню **Музыка | Добавить источник** можно добавить собрания файлов музыки, расположенных на сетевых компьюте-

рах. Добавление источников музыки осуществляется аналогично добавлению ресурса просмотра фото, описанному в предыдущем разделе.

Существует и здесь большое количество дополнений, устанавливаемых через меню **Музыка | Файлы | Аудиодополнения** (рис. 5.31).

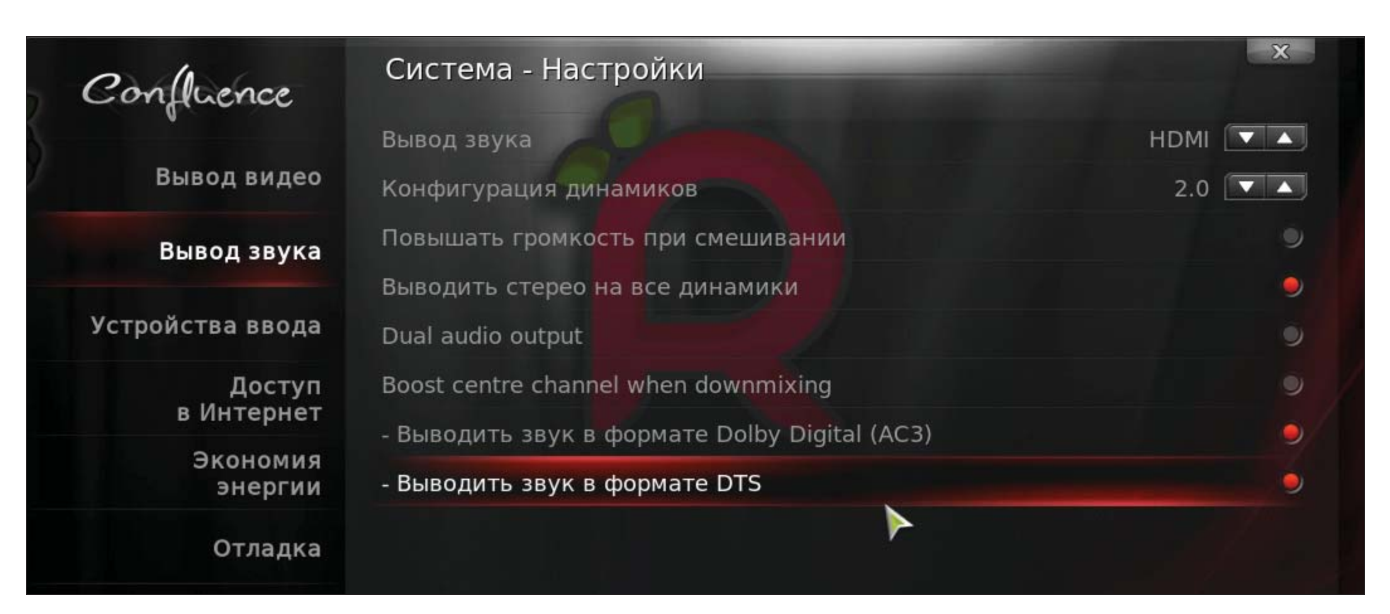


Рис. 5.30. Настройка параметров звука



Рис. 5.31. Добавление аудиодополнений

Любителям онлайнового радио предоставляется большой выбор источников, среди которых можно выделить плагины сервисов **101.RU**, **Moskva.FM**, **Online.fm** (рис. 5.32), предлагающих к прослушиванию музыкальные радиостанции на любой вкус.

Для любителей аудиокниг предлагаются плагины **audio.stepashka.com** и **Аудиокниги (asbook.ru)** (рис. 5.33).

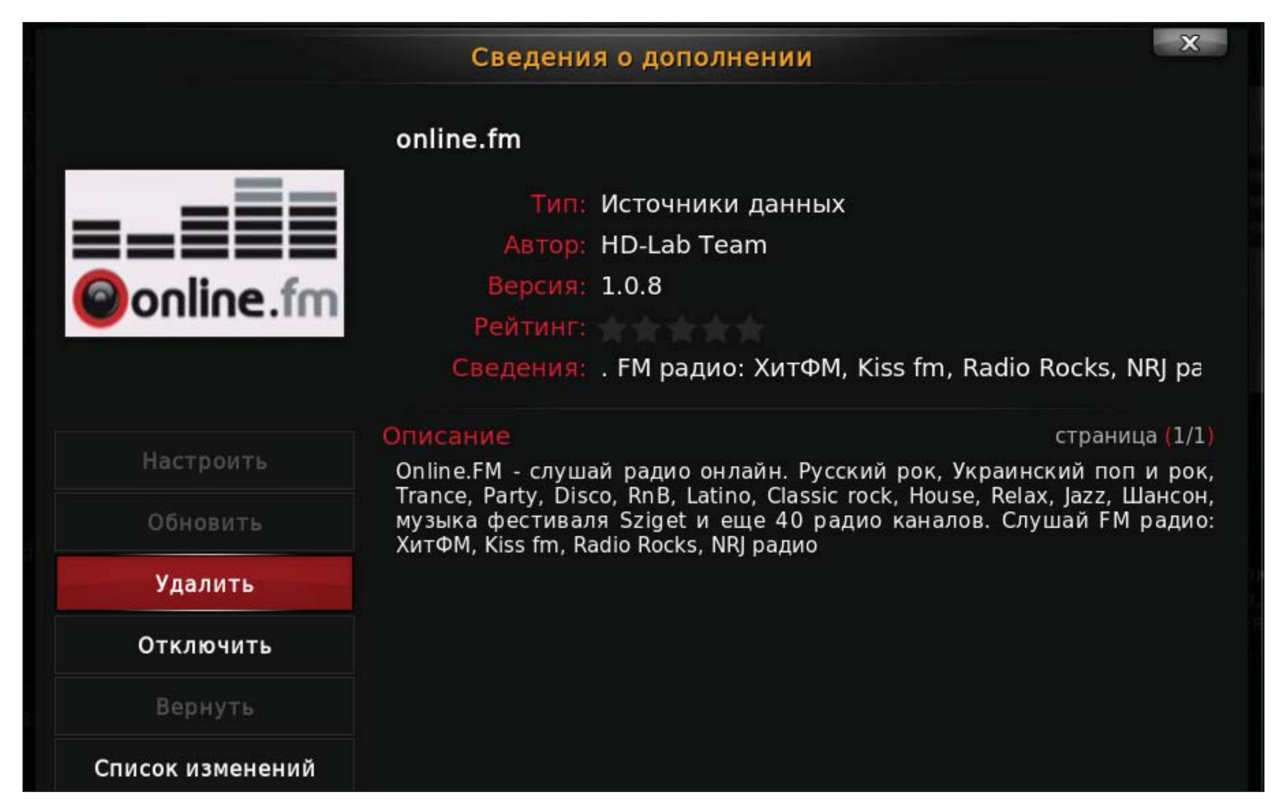


Рис. 5.32. Плагин **online.fm**

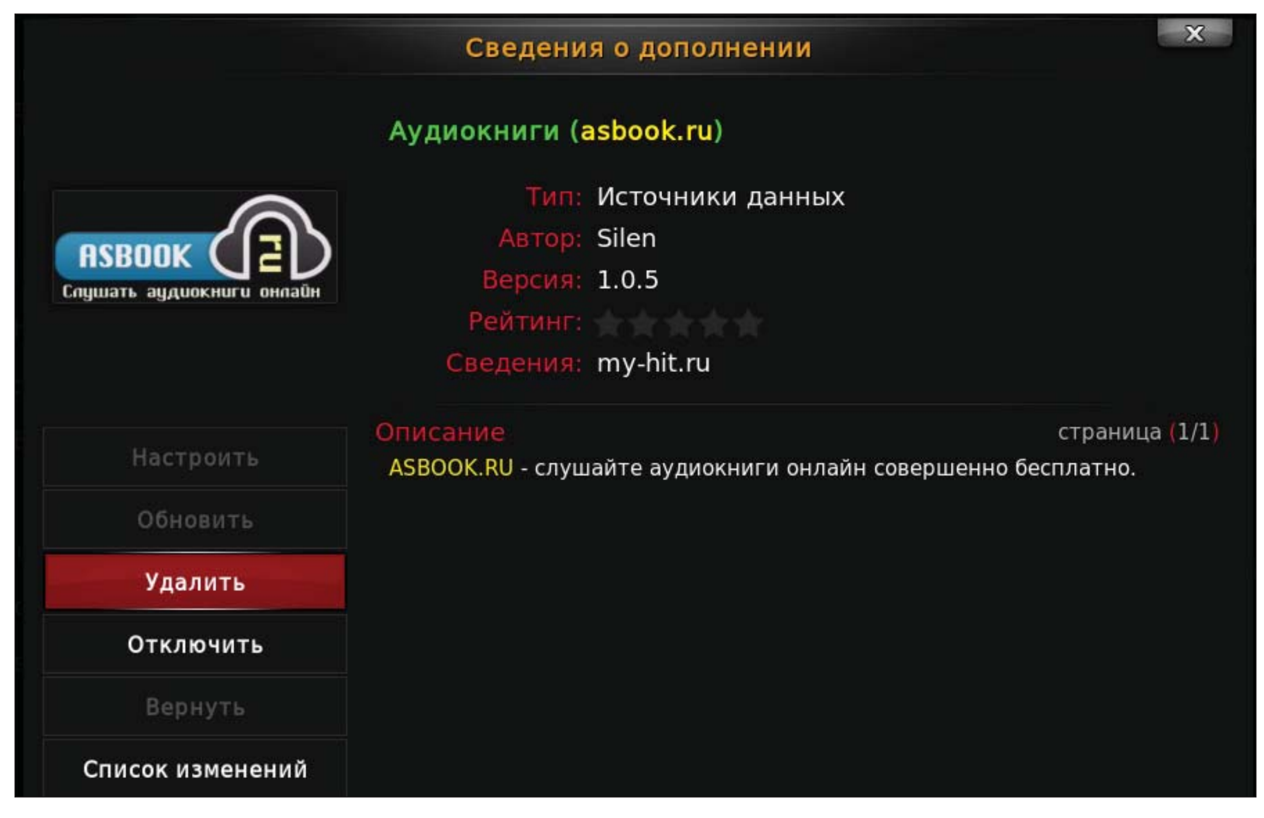


Рис. 5.33. Плагин **Аудиокниги (asbook.ru)**

5.8. Видео

XBMC поддерживает воспроизведение видео следующих форматов: AVI, MPEG, WMV, ASF, FLV, MKV, MOV, MP4, M4A, AAC, OGG, OGM, RealMedia RAM/RM/RV/RA/RMVB, 3gp, VIVO, PVA, NUV, NSV, NSA, FLI, FLC, DVR-MS, MPEG-1, MPEG-2, h.263, MPEG-4 SP и ASP, MPEG-4 AVC (h.264), HuffYUV, Indeo, MJPEG, RealVideo, RMVB, QuickTime, Sorenson, WMV, Cinepak, а также воспроизводит диски CD/DVD/Blu-Ray.

Чтобы смотреть на Raspberry Pi копии DVD-дисков или файлы, записанные с помощью Windows Media Center, понадобится декодер MPEG2, который можно приобрести у Raspberry Pi Foundation за 2,4 фунта стерлингов. Raspberry Pi Foundation также предоставляет ключ для расшифровки файлов, кодированных VC1. Вы можете приобрести и его, если он вам необходим. Ключ привязан к серийному номеру Raspberry Pi, поэтому, если вы имеет несколько таких компьютеров, придется приобрести отдельный ключ для каждого.

Для установки ключа в Raspbmc необходимо открыть раздел **Система | Настройки | Программы**, загрузить программу **Raspbmc settings**, затем на вкладке **System configuration** открыть опцию **Advanced settings**, щелкнуть на строке **VC1 codec key** и ввести ключ, полученный по электронной почте (рис. 5.34).

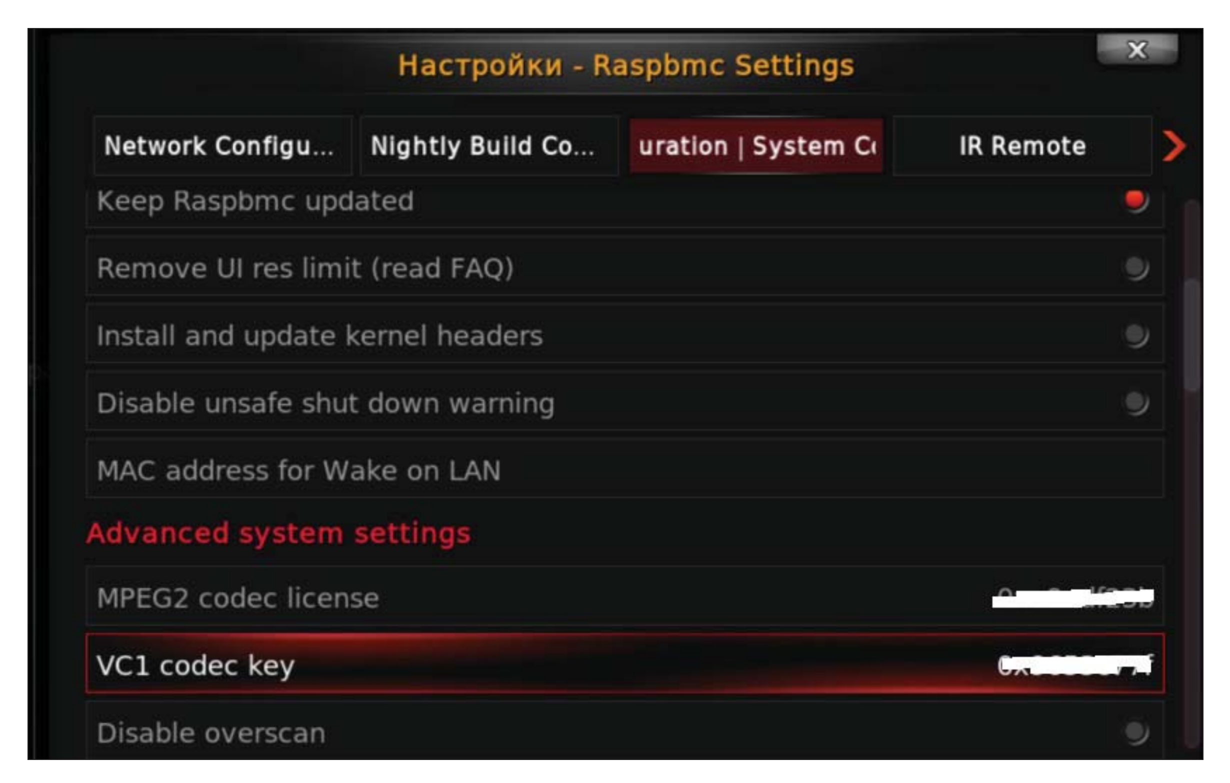


Рис. 5.34. Установка ключа для расшифровки файлов, кодированных VC1

Теперь, имея нужные кодеки, можно просматривать файлы, кодированные h.264 и MPEG2, то есть смотреть телепередачи, записанные с помощью Windows Media Center (к сожалению, только те, что без защиты от копирования). Также можно записать видеофайлы на USB-устройство (флешку или внешний диск), подключить его непосредственно к Raspberry Pi и смотреть записанные телепередачи через

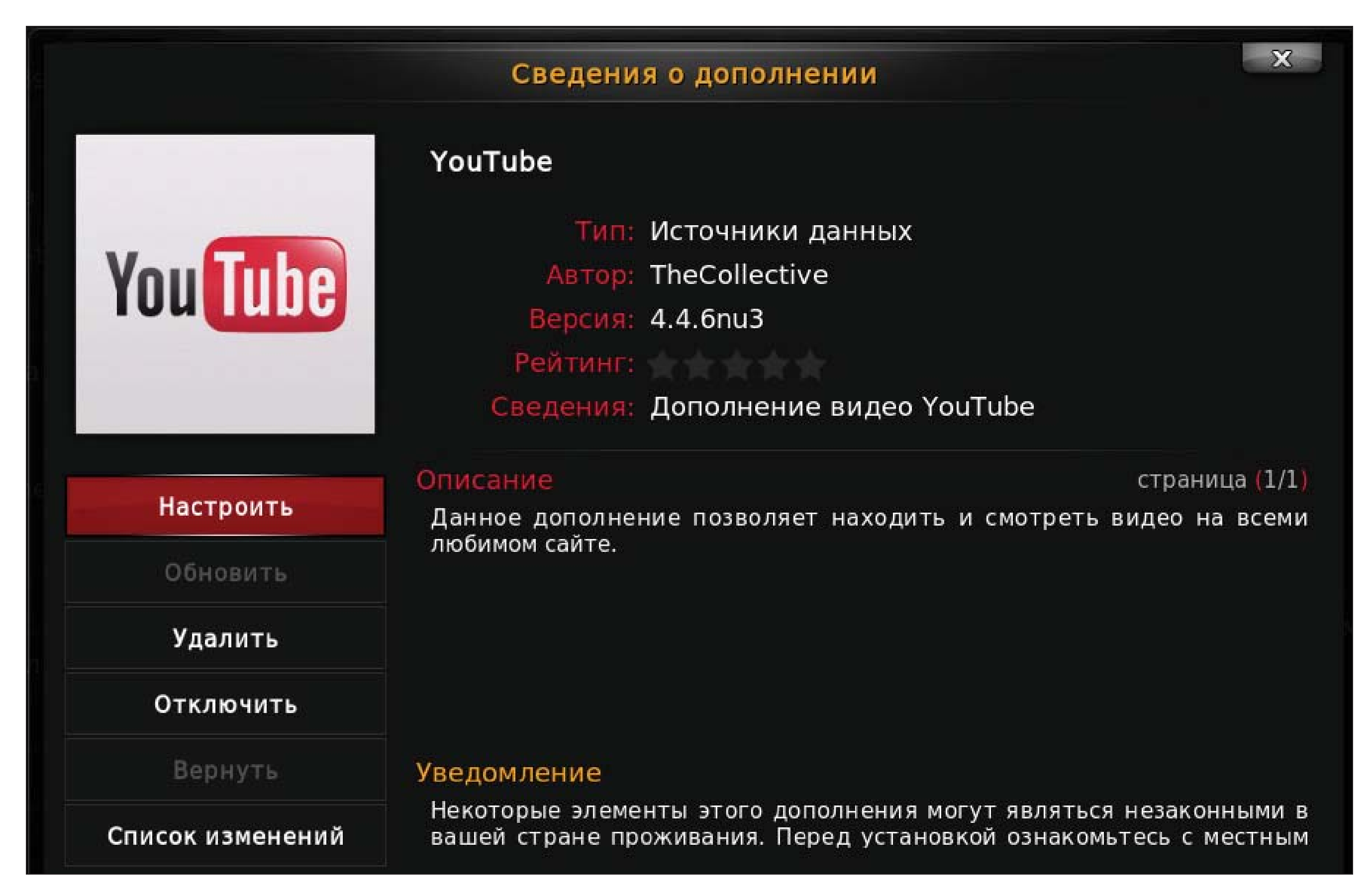


Рис. 5.35. Плагин **YouTube**

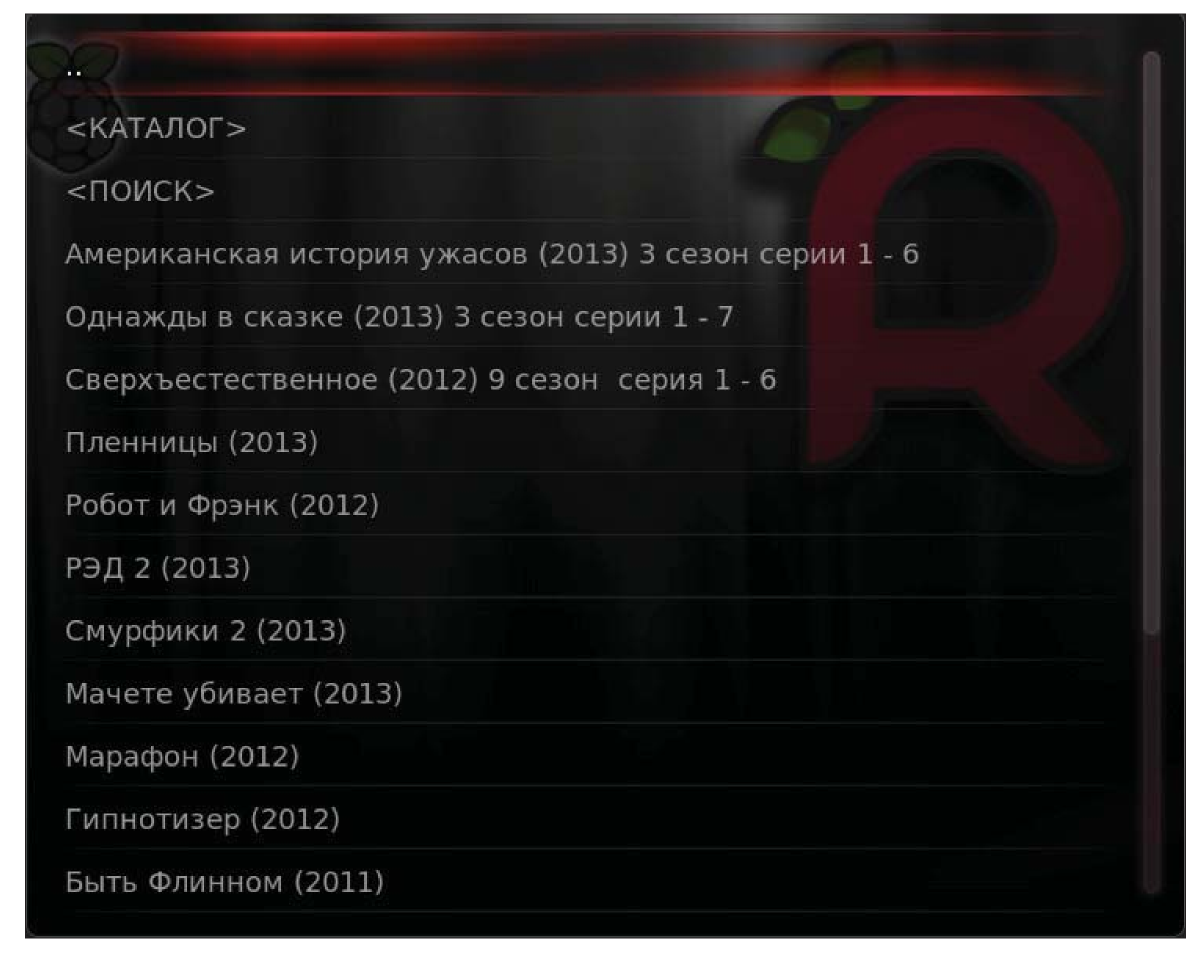


Рис. 5.36. Плагин **New-Kino.net**

Raspberry Pi. Напомню только, что подключать USB-диск к Raspberry Pi следует через USB-концентратор с внешним питанием.

Количество дополнений к системе просмотра видео огромно и на любой вкус. Установка дополнений осуществляется через меню **Видео | Дополнения**. Конечно же, стоит установить **YouTube** (рис. 5.35).

Кроме того, вас наверняка заинтересуют русскоязычные плагины из репозитория **seppius**: **New-Kino.net** (рис. 5.36), где постоянно представлены киноновинки, ресурс **Zoomby** — один из крупнейших развлекательных порталов российского Интернета, где можно онлайн, легально и бесплатно смотреть фильмы, сериалы, новости, музыкальные клипы, шоу и другие телевизионные передачи, а также ресурс **Онлайн ТВ+Архивы (telepoisk.com)** для просмотра онлайн большого количества российских каналов (рис. 5.37).

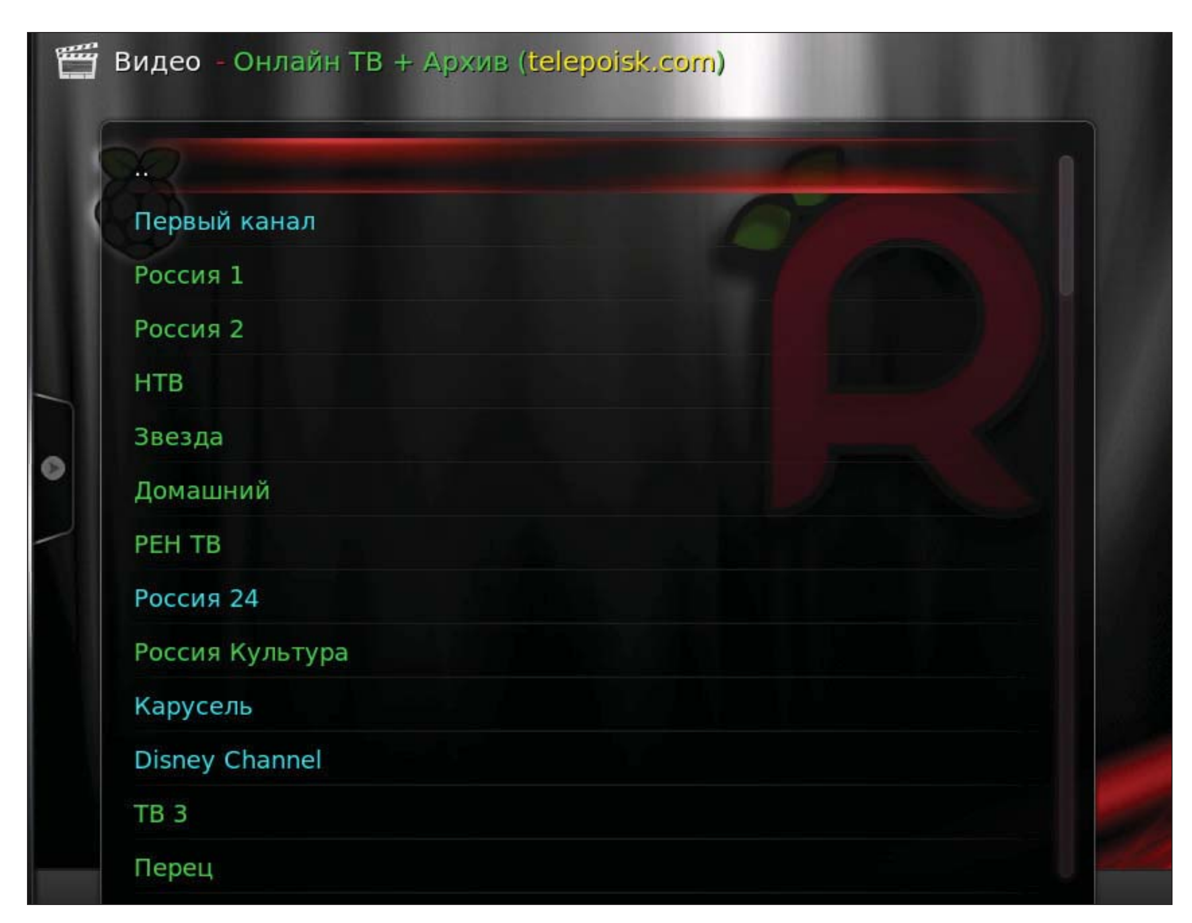


Рис. 5.37. Плагин **Онлайн ТВ+Архивы (telepoisk.com)**

5.9. Программы

Раздел **Программы** системы Raspbmc содержит дополнения, выполняющие те или иные программные функции. Можно выбрать, например, дополнение **Gmail**, которое позволяет просмотреть почту в ящике на **gmail.com** (рис. 5.38) или **Arora Web Browser** для серфинга по сети Интернет. В общем, смотрите, устанавливайте, пробуйте.

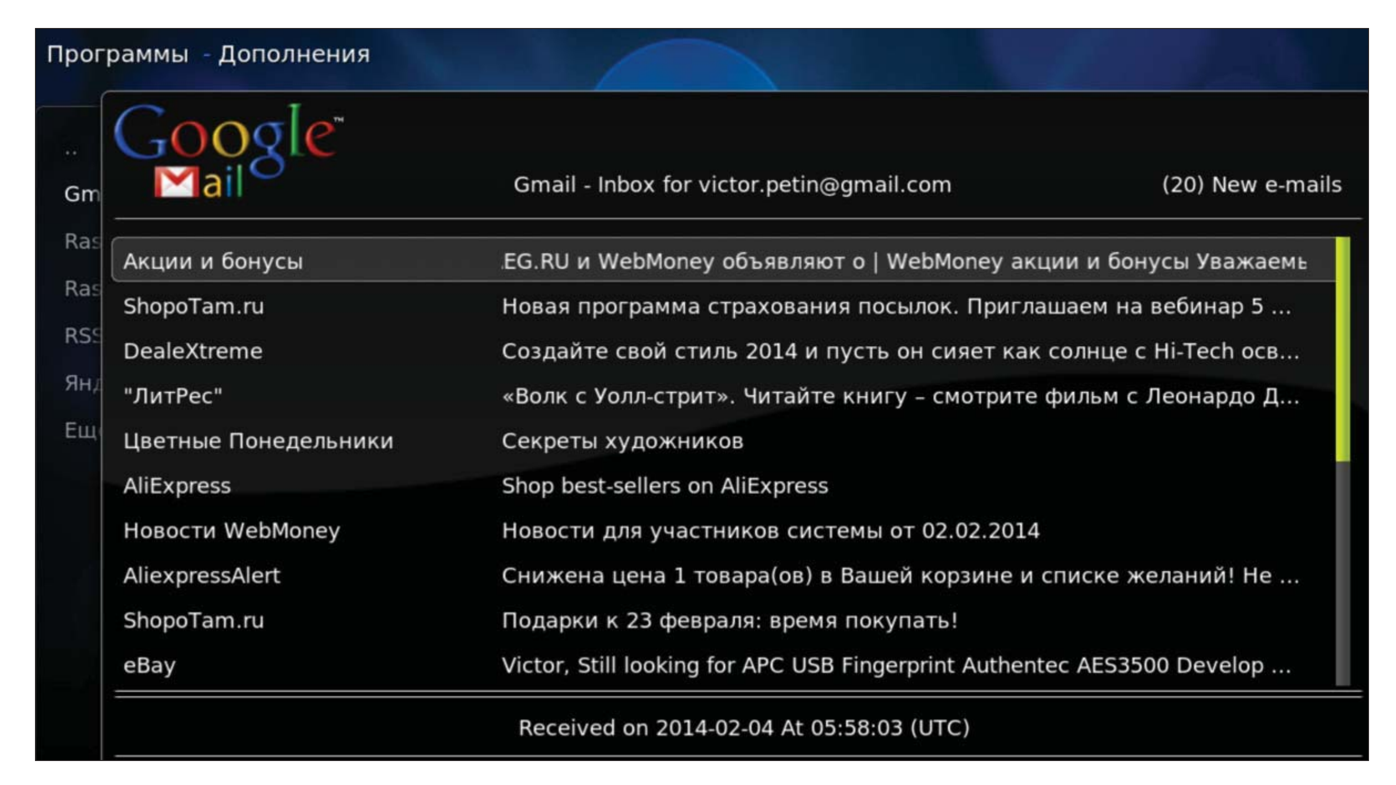


Рис. 5.38. Плагин **Gmail Checker** для просмотра электронной почты

5.10. Разгон системы

Для разгона системы Raspbmc на Raspberry Pi запускаем программу **Raspbmc Settings**, в меню **System Configuration** выбираем пункт **CPU overclock** (рис. 5.39)

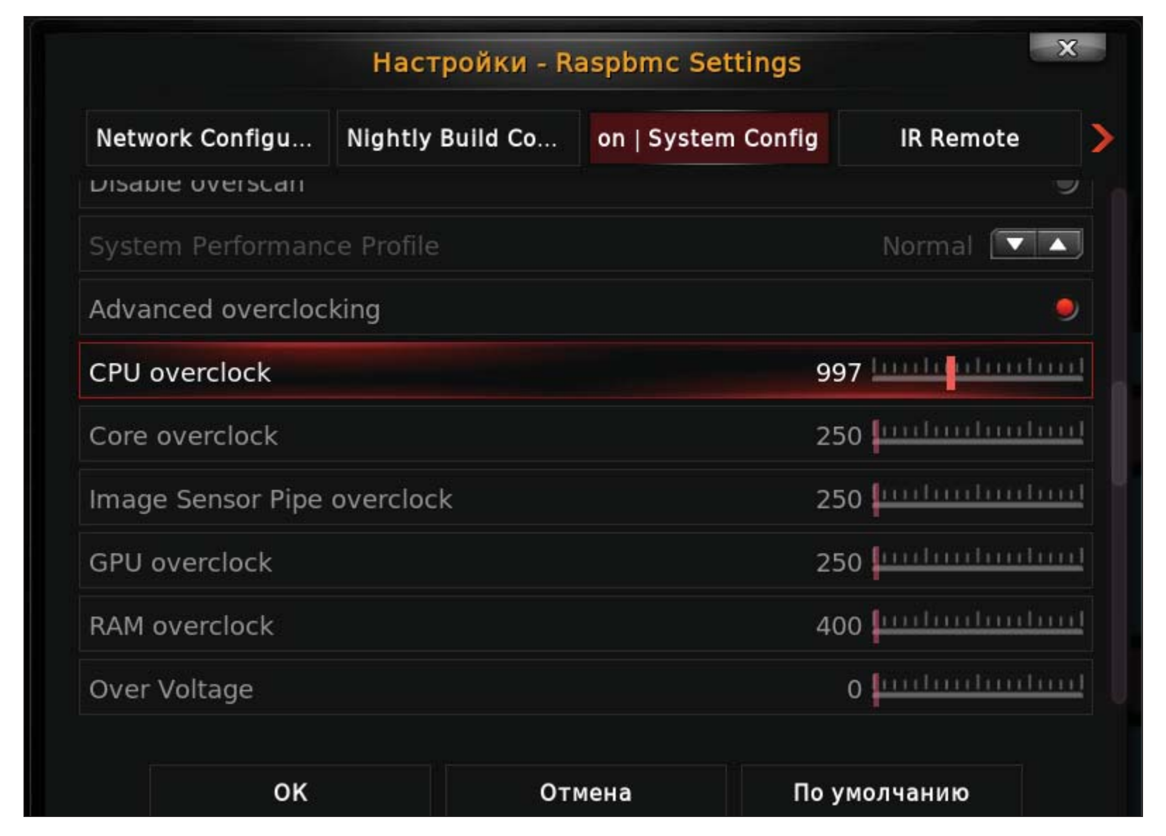


Рис. 5.39. Установка параметра **CPU overclock**

и выставляем частоту 1000 МГц. Сохраняем. При сохранении система перезагружается.

На частоте 1000 Мгц Raspbmc не запустилась... Для выхода из столь неприятной ситуации требуется править ее файл config.txt, а чтобы добраться до него (см. *главу 2*), при запуске нажимаем клавишу <Shift>, попадаем в меню выбора операционных систем NOOBS, выбираем **Raspbmc** и затем пункт **Edit config**. Выставляем параметр `arm_freq=900`. Перезагружаемся. Теперь все нормально.

5.11. Управление Raspberry Pi на ОС Raspbmc с помощью планшета Android

Разработчики XBMC предлагают в интернет-магазинах программного обеспечения Google Play Store и Apple App Store бесплатные приложения для дистанционного управления устройством Raspberry Pi с планшетов и смартфонов. Программа Official XBMC Remote будет особенно интересна тем, кто дополнил свой архив фильмов и ТВ-сериалов картинками с обложек и кадрами — она показывает не только списки с названиями кинолент, но и данные изображения, а поиск ТВ-серий осуществляет по их баннерам.

Мини-ПК Raspberry Pi на ОС Raspbmc можно подключить к мобильному устройству только в том случае, если оба устройства находятся в одной и той же локальной сети, а на Raspbmc запущен специальный сетевой сервер. Чтобы его настроить, зайдем в меню **Система | Настройки | Службы | Веб-сервер**, включим там опцию **Разрешить управление XBMC по HTTP** и зададим имя пользователя и пароль. В поле **Порт** вводим значение `8080` (рис. 5.40). Кроме того, находим IP-адрес нашего Raspberry Pi (**Система | Сведения о системе | Сеть**) — он понадобится нам при последующих настройках.

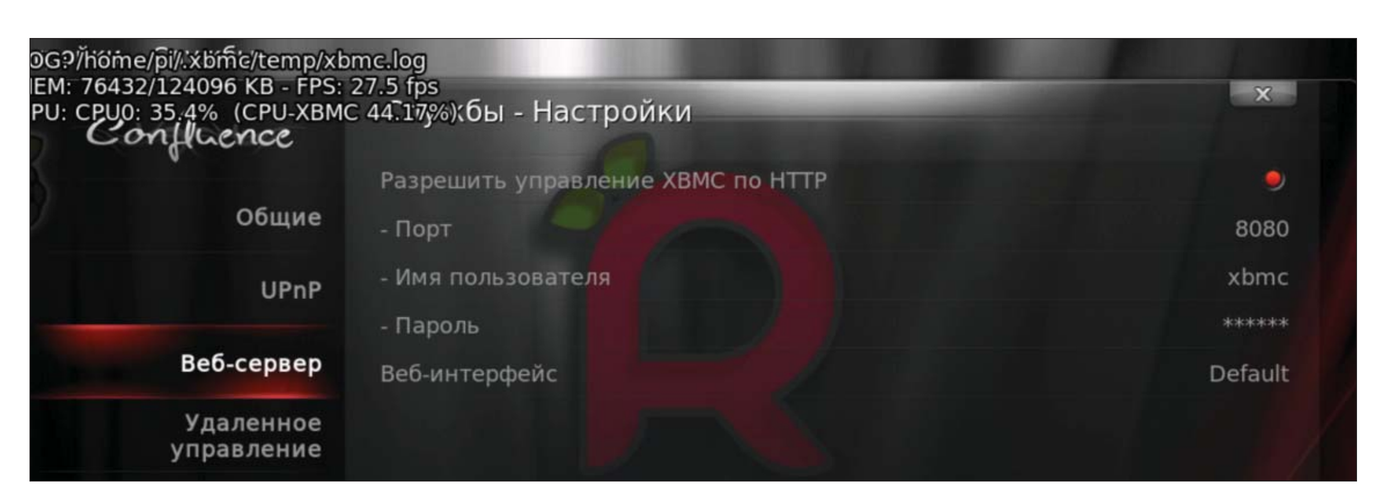


Рис. 5.40. Настройки Raspbmc для удаленного управления: опция **Порт**

В пункте **UPnP** разрешим **управление XBMC по UPnP** (рис. 5.41).

В пункте **Удаленное управление** разрешим **программам на других системах управлять XBMC** (рис. 5.42).

Скачиваем на планшет/смартфон приложение Official XBMC Remote (рис. 5.43).

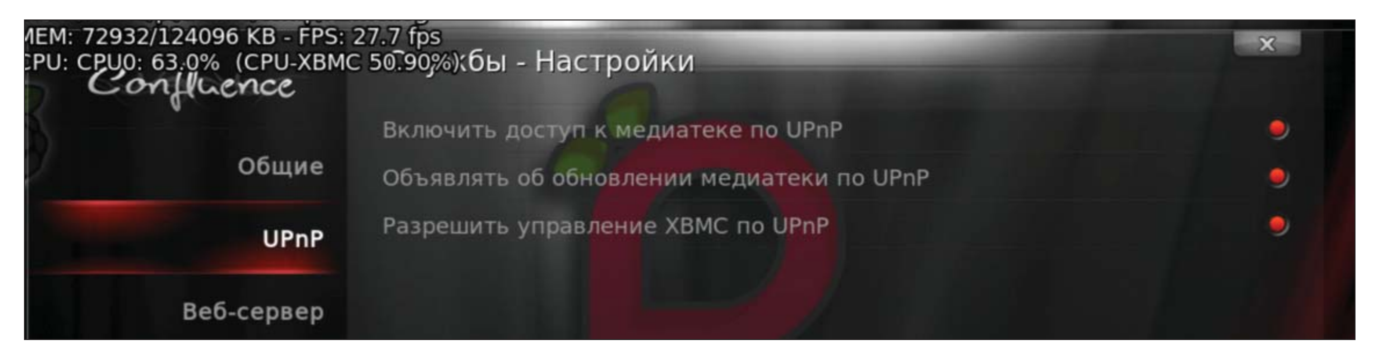


Рис. 5.41. Настройки Raspbmc для удаленного управления: опция **UPnP**

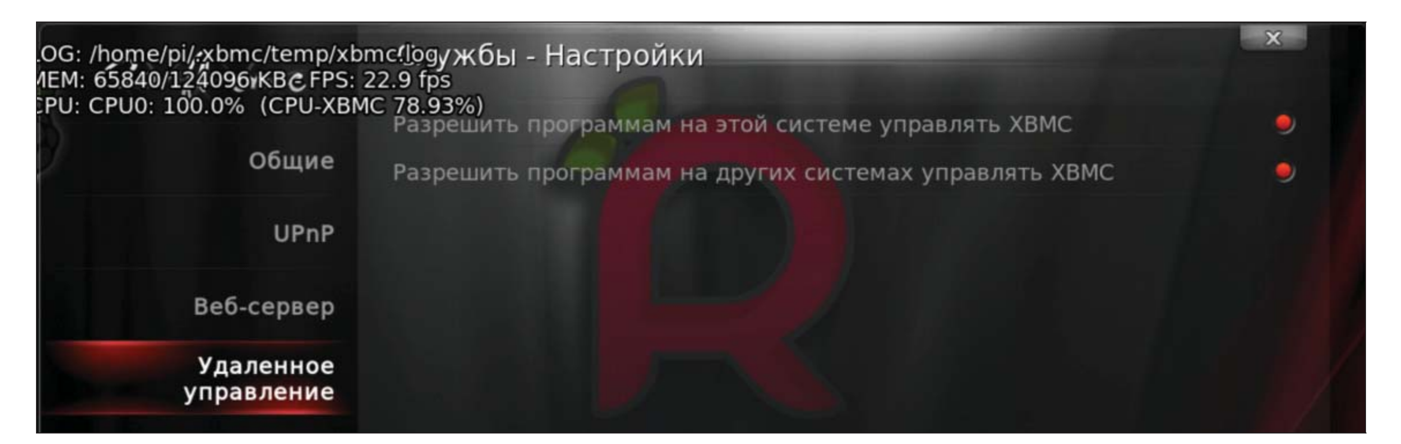


Рис. 5.42. Настройки Raspbmc для удаленного управления: опция **Удаленное управление**

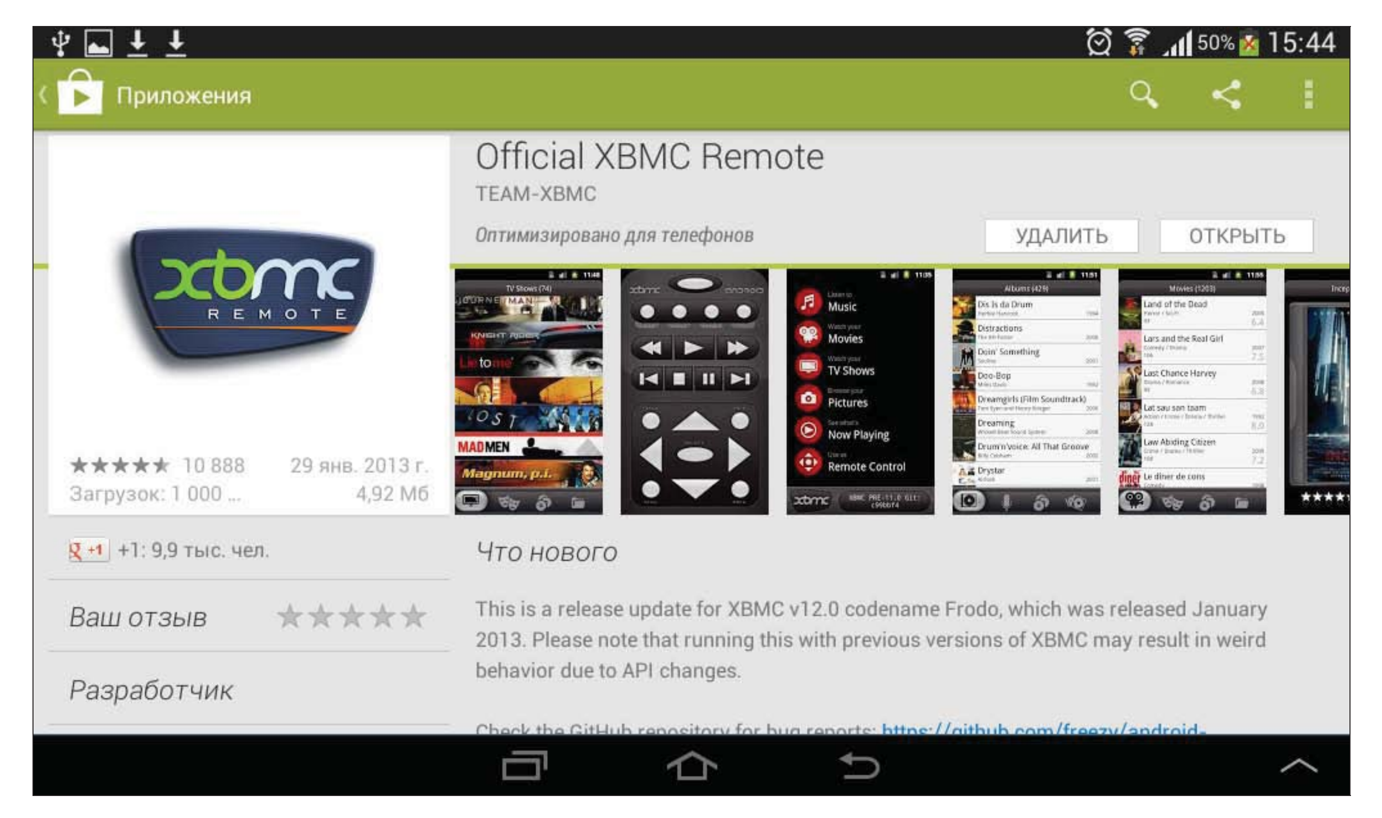


Рис. 5.43. Приложение **Official XBMC Remote** в Google Play Store

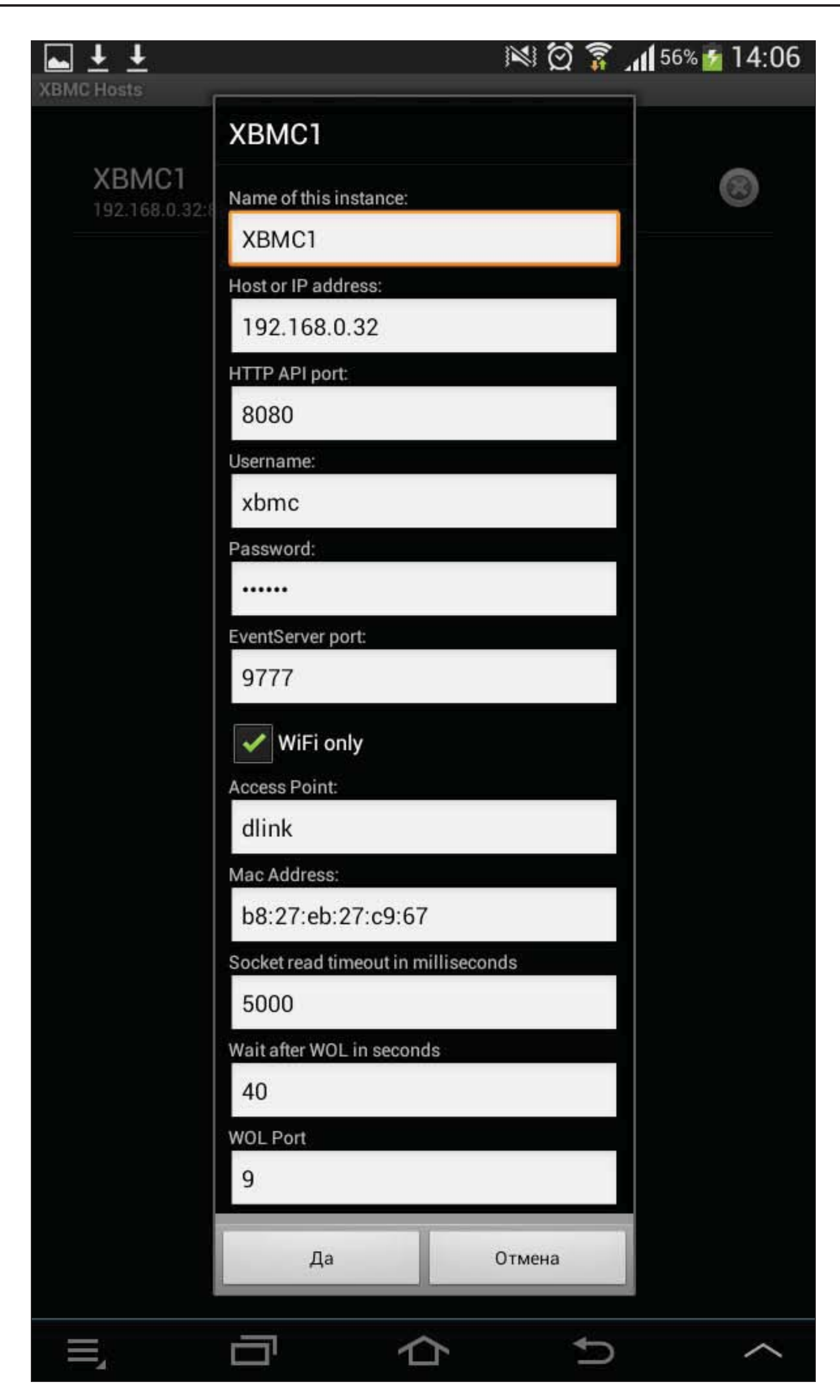


Рис. 5.44. Добавление настроек XBMC

В приложении Official XBMC Remote заходим в настройки (**Settings | Manage | XBMC Hosts**) и добавляем данные нашего Raspberry Pi (рис. 5.44).

Подключаемся с мобильного устройства к сети, где расположен наш Raspberry Pi, и получаем возможность управлять с него плеером на Raspberry Pi. Из главного меню плеера на планшете/смартфоне (рис. 5.44) можно либо выбрать удаленное управление (пункт **Remote Control**) с помощью кнопок или подобия тачпада (рис. 5.46), либо открывать картинки, музыку и видео из списков (пункты **Music**, **Movies**, **TV Shows**, **Pictures**). Интерфейс приложения очень удобен, а из недостатков — отсутствие стриминга (воспроизведения медиафайлов XBMC на экране мобильного устройства).

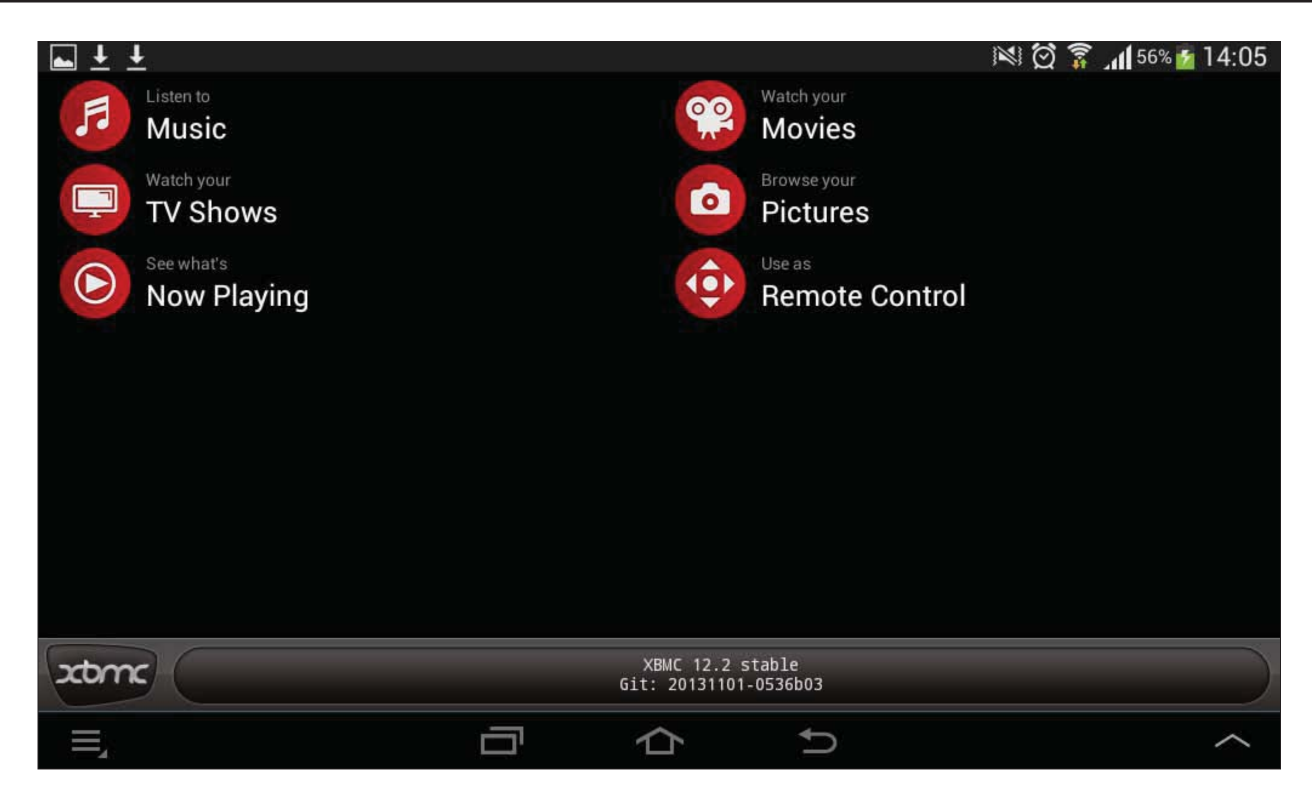


Рис. 5.45. Добавление настроек XBMC



Рис. 5.46. Инструменты удаленного управления Raspberry Pi на планшете

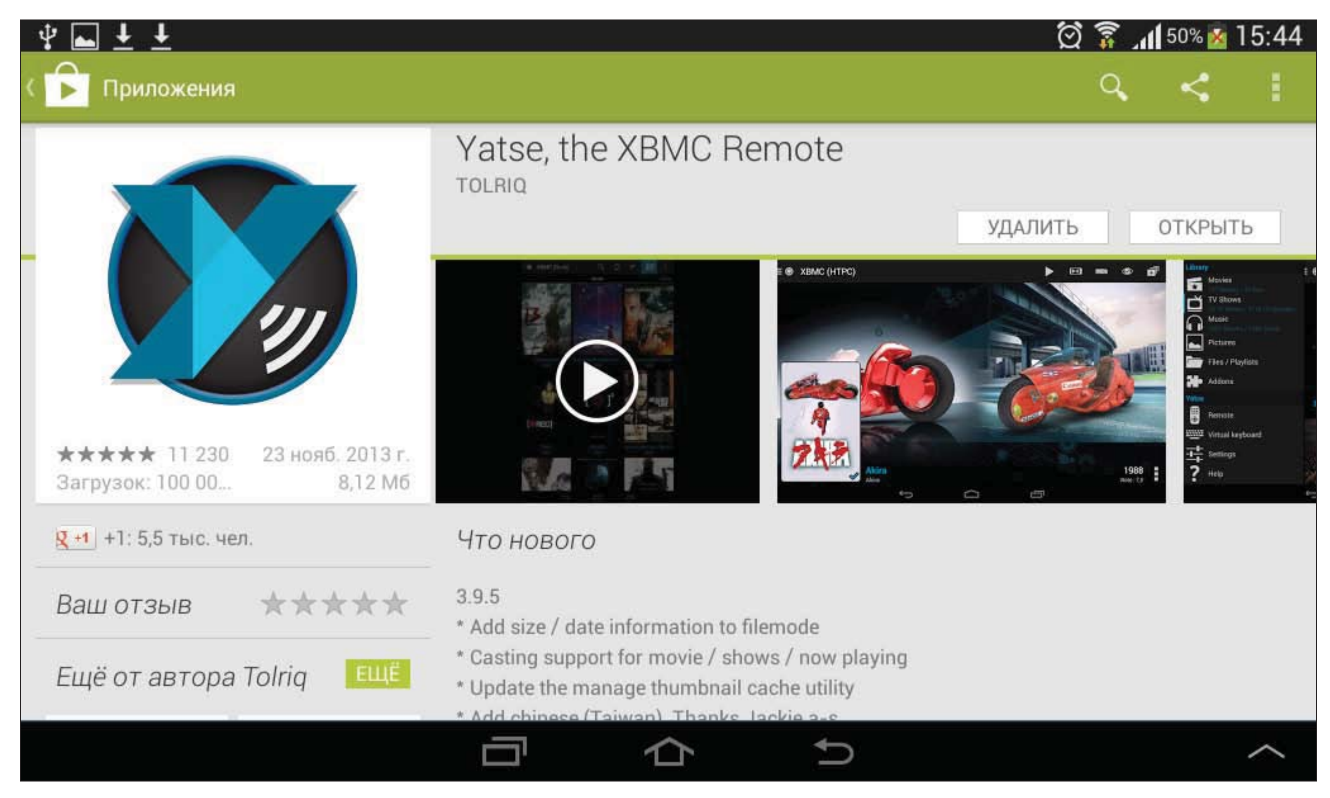


Рис. 5.47. Приложение **Yatse** в Google Play Store

Гораздо более мощной программой управления XBMC для Android-устройств признана Yatse (рис. 5.47).

Ее основные функции:

- голосовые команды;
- офлайн-синхронизация библиотек;
- современный и понятный интерфейс Holo;
- поддержка многочисленных установок XMBC;
- многочисленные виджеты для быстрых дистанционных действий, деталей воспроизводимого в данный момент медиа или списка последних файлов;
- разные настраиваемые виды;
- сохранение в облаке хостов и настроек для использования на разных устройствах;
- потоковая передача ваших видео на устройствах Android или UPnP/AirPlay;
- отправка медиафайлов Android на устройства Xbmc или UPnP/AirPlay.

И особенности:

- расширение DashClock;
- поддержка всех медиа XBMC;
- поддержка списков воспроизведения и файлов;
- поддержка всех дистанционных команд XBMC с помощью оптимизированных экранов или виджетов;

- поддержка уведомлений Android для текущего воспроизведения;
- поддержка управления/виджетов на экране блокировки Android;
- быстрое обнаружение медиа с помощью поиска;
- сортировка медиа по предпочтениям;
- многочисленные привлекательные режимы просмотра списков ваших медиафайлов;
- переключение статуса просмотренных медиа;
- детализация текущего воспроизведения с выбором субтитров и аудиопотоков;
- поддержка пробуждения Wake on Lan (WOL) ;
- изменение текущих обоев на фан-арт или эскиз текущего воспроизведения;
- различные плагины для СМС-вызовов и уведомлений Android;
- отправка медиа на XBMC из YouTube или браузера;
- кнопка IMDB для дополнительных деталей медиа, если доступно;
- поддержка постеров и баннеров телешоу;
- поддержка возобновления медиа;
- оптимизация скорости даже на старых телефонах;
- оптимизация для низкого энергопотребления;
- поддержка тем при покупке ключа разблокировки;
- интерфейс для управления Xbmc/Yatse из других приложений Android.

Функция стриминга доступна только в платной версии — для этого необходимо скачать программу Yatse Remote, стоимость которой 120 рублей. Кроме того, потребуется на Raspbmc в меню **Система | Настройки | Службы | Zeroconf** включить опцию **Объявлять эти сервисы другим системам по Zeroconf** (рис. 5.48), а также установить Yatse Stream Plugin из Google.

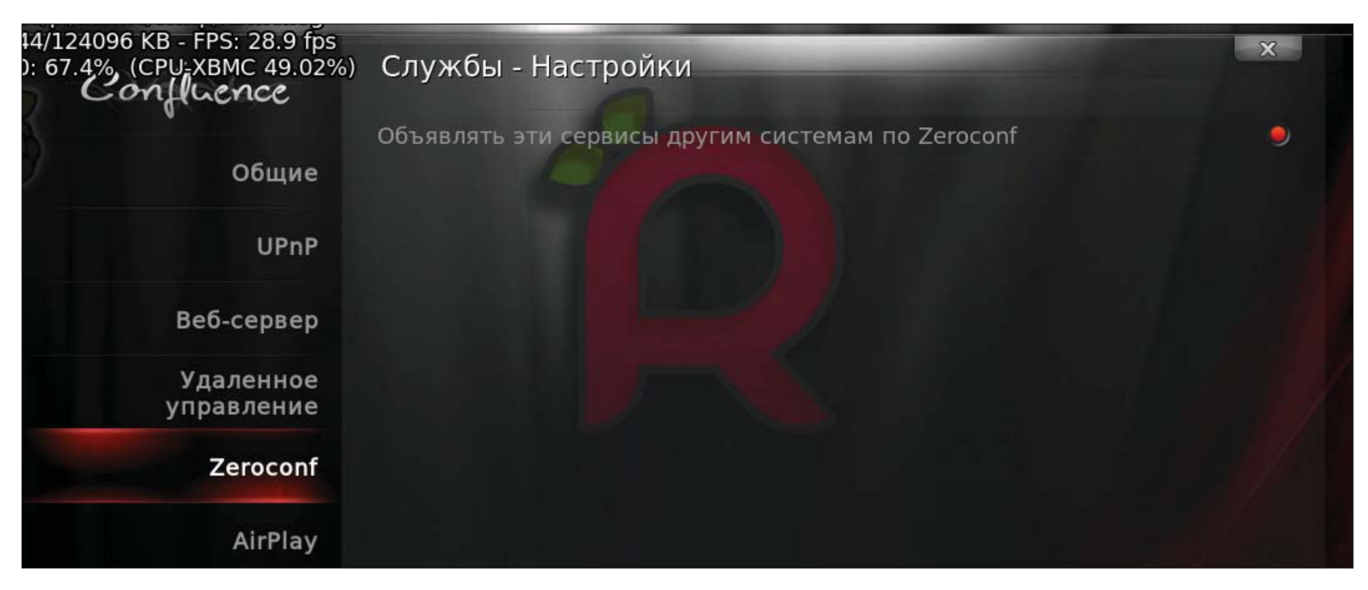


Рис. 5.48. Настройки XBMC для стриминга

5.12. Управление Raspbmc с помощью пульта

У Raspberry Pi есть особенность, которой может позавидовать любой неттоп и подавляющее большинство медиаплееров, — это поддержка технологии CEC (Consumer Electronics Control). Протокол CEC позволяет отправлять команды пульта дистанционного управления от одного устройства другому — т. е. дает возможность передавать команды от телевизионного пульта медиацентру XBMC. А также использовать экспериментальные функции XBMC типа "пауза эфира" и осуществлять запись эфирных программ, транслирующихся серверами типа TVHeadend, DVBLink, MediaPortal и MythTV. Большинство телевизоров, выпущенных за последние пару лет, поддерживают эту технологию.

Если у вас нет такого телевизора или вы обходитесь монитором, можно воспользоваться для управления Raspbmc любым телевизионным пультом. Нам потребуется лишь подключить к выводам GPIO Raspberry Pi какой-либо ИК-ресивер. Этот вопрос мы рассмотрим в *разд. 6.1.5* нашей книги.

5.13. Написание плагина для Raspbmc

Рассмотрим процесс создания собственного плагина для XBMC. Как известно, виджеты для XBMC пишутся на языке Python. Python в XBMC полнофункциональный, присутствует почти вся стандартная библиотека. Для взаимодействия с XBMC добавлены 5 модулей: xbmc, xbmcgui, xbmcplugin, xbmcaddon и xbmcvfs, которые вместе составляют XBMC Python API. Краткую справку по модулям языка Python можно найти на сайте **http://mirrors.xbmc.org/docs/python-docs/**.

Изначально разработчики XBMC применяли термин "аддон" (addon) для любых дополнений, которые, в свою очередь, подразделяются на плагины (источники контента), скрипты (программы), скреперы (загрузчики информации к медиаконтенту) и прочие. Однако со временем понятия "аддон" и "плагин" перепутались, и сейчас их часто используют для обозначения любого дополнения.

5.13.1. Немного теории, необходимой для написания простого плагина

Рассмотрим некоторые базовые функции библиотек XBMC языка Python, необходимых нам для создания собственного плагина (см. *разд. 5.13.3*).

Сначала подключаем необходимые нам библиотеки.

```
import  xbmc, xbmcgui
```

После этого создадим простой класс:

```
class MyClass(xbmcgui.Window):
  print 'hello world'
```

После инициализации объекта класса метод `DoModal` предоставляет возможность видеть графическое окно, пока мы его не закроем:

```
mydisplay = MyClass()
mydisplay .doModal()
del mydisplay
```

Если запустить этот скрипт, то мы увидим пустое окно (оператор `print` вместо консоли пишет в лог XBMC), из которого можно выйти только перезагрузкой XBMC. Поэтому надо предусмотреть выход из класса (листинг 6.1).

Листинг 6.1. Скрипт MyClass

```
import xbmc, xbmcgui
# actioncodes получить из keymap.xml
ACTION_PREVIOUS_MENU = 10

class MyClass(xbmcgui.Window):
  def onAction(self, action):
    if action == ACTION_PREVIOUS_MENU:
      self.close()
mydisplay = MyClass()
mydisplay .doModal()
del mydisplay
```

Каждый раз когда мы нажимаем на кнопку **BACK** (или клавишу <Esc>), то выходим из класса.

Теперь время показать в созданном нами окне текст. Добавим текстовую метку `Label` и воспользуемся функцией `ControlLabel`. Эта функция содержит в себе параметры: позиция, цвет, прозрачность и размер шрифта.

```
self.strAction = xbmcgui.ControlLabel(300, 520, 200, 200, ' ', 'font14',
'0xFFFFFF00')
self.addControl(self.strAction)
self.strAction.setLabel('BACK to quit')
```

Теперь удалим текстовую метку с помощью функции `removeControl`, пользуясь, чтобы удалить надпись, клавишей <Backspace> на клавиатуре (листинг 6.2).

Листинг 6.2. Скрипт MyClass с опцией удаления метки

```
import xbmc, xbmcgui

#get actioncodes from keymap.xml
ACTION_PREVIOUS_MENU = 10
ACTION_SELECT_ITEM = 7
ACTION_PARENT_DIR = 9

class MyClass(xbmcgui.Window):
  def onAction(self, action):
    if action == ACTION_PREVIOUS_MENU:
      self.close()
```

```
    if action == ACTION_SELECT_ITEM:
      self.strAction = xbmcgui.ControlLabel(300, 200, 200, 200, '', 'font14',
'0xFF00FF00')
      self.addControl(self.strAction)
      self.strAction.setLabel('Hello world')
    if action == ACTION_PARENT_DIR:
      self.removeControl(self.strAction)

mydisplay = MyClass()
mydisplay .doModal()
del mydisplay
```

Для создания кнопки воспользуемся функцией `ControlButton`:

```
self.button0 = xbmcgui.ControlButton(350, 500, 80, 30, "HELLO")
self.addControl(self.button0)
self.setFocus(self.button0)
```

Определим действие при нажатии на кнопку:

```
def onControl(self, control):
if control == self.button0:
 print 'button pushed'
```

Можно добавить возможность ввода текста с помощью виртуальной клавиатуры (листинг 6.3).

Листинг 6.3. Скрипт MyClass с добавленной кнопкой

```
import xbmc, xbmcgui

ACTION_PREVIOUS_MENU = 10

class MyClass(xbmcgui.Window):
  def __init__(self):
    self.strActionInfo = xbmcgui.ControlLabel(100, 120, 200, 200, '', 'font13',
                                                                   '0xFFFF00FF')
    self.addControl(self.strActionInfo)
    self.strActionInfo.setLabel('Push BACK to quit')
    self.strActionInfo = xbmcgui.ControlLabel(100, 300, 200, 200, '', 'font13',
                                                                   '0xFFFFFFFF')
    self.addControl(self.strActionInfo)
    keyboard = xbmc.Keyboard('mytext')
    keyboard.doModal()
    if (keyboard.isConfirmed()):
      self.strActionInfo.setLabel(keyboard.getText())
    else:
      self.strActionInfo.setLabel('user canceled')
```

```
  def onAction(self, action):
    if action == ACTION_PREVIOUS_MENU:
      self.close()

mydisplay = MyClass()
mydisplay .doModal()
del mydisplay
```

Очень часто необходимо использовать списки. Код создания списка представлен в листинге 6.4.

Листинг 6.4. Код создания списка

```
import xbmc, xbmcgui

ACTION_PREVIOUS_MENU = 10

class MyClass(xbmcgui.Window):
  def __init__(self):
    self.strActionInfo = xbmcgui.ControlLabel(250, 80, 200, 200, '', 'font14',
'0xFFBBBBFF')
    self.addControl(self.strActionInfo)
    self.strActionInfo.setLabel('Push BACK to quit')
    self.list = xbmcgui.ControlList(200, 150, 300, 400)
    self.addControl(self.list)
    self.list.addItem('Item 1')
    self.list.addItem('Item 2')
    self.list.addItem('Item 3')
    self.setFocus(self.list)

  def onAction(self, action):
    if action == ACTION_PREVIOUS_MENU:
      self.close()

  def onControl(self, control):
    if control == self.list:
      item = self.list.getSelectedItem()
      self.message('You selected : ' + item.getLabel())

  def message(self, message):
    dialog = xbmcgui.Dialog()
    dialog.ok(" My message title", message)

mydisplay = MyClass()
mydisplay.doModal()
del mydisplay
```

Если не хватает места в главном окне, можно создать дочернее окно. При этом для него создается еще один класс (листинг 6.5).

Листинг 6.5. Класс для дочернего окна

```
import xbmc, xbmcgui

ACTION_PREVIOUS_MENU = 10
ACTION_SELECT_ITEM = 7

class MainClass(xbmcgui.Window):
  def __init__(self):
    self.strActionInfo = xbmcgui.ControlLabel(180, 60, 200, 200, '', 'font14',
'0xFFBBBBFF')
    self.addControl(self.strActionInfo)
    self.strActionInfo.setLabel('Push BACK to quit - A to open another window')
    self.strActionInfo = xbmcgui.ControlLabel(240, 250, 200, 200, '', 'font13',
'0xFFFFFFFF')
    self.addControl(self.strActionInfo)
    self.strActionInfo.setLabel('This is the first window')

  def onAction(self, action):
    if action == ACTION_PREVIOUS_MENU:
      self.close()
    if action == ACTION_SELECT_ITEM:
      popup = ChildClass()
      popup .doModal()
      del popup

class ChildClass(xbmcgui.Window):
  def __init__(self):
    self.addControl(xbmcgui.ControlImage(0,0,800,600, 'background.png'))
    self.strActionInfo = xbmcgui.ControlLabel(200, 60, 200, 200, '', 'font14',
'0xFFBBFFBB')
    self.addControl(self.strActionInfo)
    self.strActionInfo.setLabel('Push BACK to return to the first window')
    self.strActionInfo = xbmcgui.ControlLabel(240, 200, 200, 200, '', 'font13',
'0xFFFFFF99')
    self.addControl(self.strActionInfo)
    self.strActionInfo.setLabel('This is the child window')

  def onAction(self, action):
    if action == ACTION_PREVIOUS_MENU:
      self.close()

mydisplay = MainClass()
mydisplay .doModal()
del mydisplay
```

5.13.2. Структура простого плагина

Рекомендации по разрабортке плагинов можно найти на сайте **http://wiki.xbmc.org/index.php?title=Add-on_development**.

Для плагина создается каталог с названием <addonType>[.<mediaType>].<yourPluginName>. Структура этого каталога должна быть следующей:

```
addon.py
addon.xml
changelog.txt
fanart.jpg
icon.png
LICENSE.txt
/resources
  /settings.xml
  /language/
  /lib/
  /media/
```

Где:

- addon.py — основной код плагина (название указывается в файле addon.xml);
- addon.xml — файл настроек плагина. Он сообщает XBMC:
  - тип плагина (видео, аудио, изображения, скрипт и т. д.);
  - какой файл надо выполнить при обращении (тот самый *.py);
  - платформу и зависимости, версию, автора и описание плагина.
- changelog.txt, LICENSE.txt — информация об изменениях и лицензии;
- fanart.jpg — фоновое изображение плагина;
- icon.png — значок плагина (256×256 пикселов);
- /resources/settings.xml — файл настроек переменных, используемых в плагине;
- /resources/language/ — языковые файлы;
- /resources/lib/ — лучшее место для хранения дополнительных библиотек Python;
- /resources/media/ — место хранения медиафайлов (изображения, звуки, видео и т. п.).

5.13.3. Проект создания плагина для получения погоды с сайта Народного мониторинга

Теперь напишем собственный плагин, который будет получать с сайта Народного мониторинга (**http://www.narodmon.ru**) список устройств, расположенных в радиусе 50 км от определенной точки, и демонстрировать показания погодных датчиков на этих устройствах. Воспользуемся для этого API сайта **narodmon.ru**.

Нам потребуются два метода API:

- `sensorNear` — запрос списка датчиков поблизости;
- `sensorDev` — запрос списка датчиков по ID устройства мониторинга.

Описание методов API **narodmon.ru** можно найти на странице **http://narodmon.ru/#apidoc**.

Для создания приложения необходимо получить ключи АР — заходим в свой профиль на сайте **narodmon.ru** (**Мои данные | Мои ключи API | запросить новый**), заполняем данные и получаем ключ (рис. 5.49).

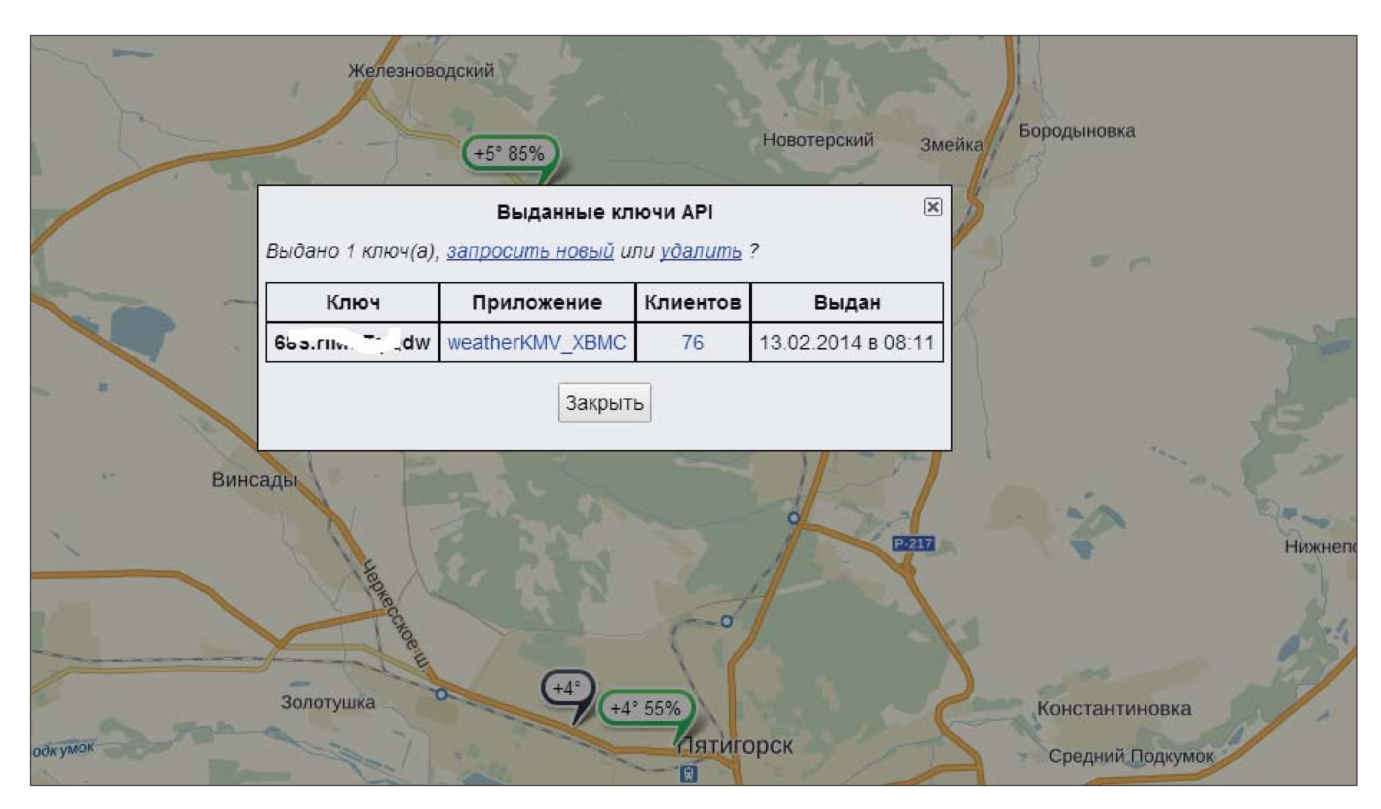


Рис. 5.49. Получение ключа API на сайте **narodmon.ru**

При запуске плагина наш скрипт обращается к API сайта **narodmon.ru** для получения списка устройств, находящихся в радиусе 50 км от выбранной точки центра региона Кавказские Минеральные Воды с координатами (lat = 44.05, lng = 42.80). Полученные результаты выводятся в окно в виде списка. При выборе элемента списка скрипт обращается к сайту для получения списка датчиков выбранного устройства и выводит список датчиков и последние показания (рис. 5.50).

Содержание файлов default.py и addon.xml представлены в листингах 6.6 и 6.7 соответственно.

Листинг 6.6. Файл default.py

```
 # -*- coding: utf-8 -*-
# Licence: GPL v.3 http://www.gnu.org/licenses/gpl.html

# Импортируем модули
import xbmcgui
```

```
import urllib,urllib2,json,re
import hashlib
import uuid
import time
from datetime import datetime

# Коды клавиатурных действий
ACTION_PREVIOUS_MENU = 10 # По умолчанию - ESC
ACTION_NAV_BACK = 92 # По умолчанию - Backspace

#  API
api_key = '68S.rlMeTgQdw'

#
app_id = str(uuid.getnode())
md5_app_id = hashlib.md5(app_id).hexdigest()

# список устройств поблизости
# данные датчиков устройства


# Главный класс-контейнер
class MyAddon(xbmcgui.Window):

    def __init__(self):
        # список устройств поблизости
        self.getDevices()
        #
        self.label0 = xbmcgui.ControlLabel(100, 50, 800, 50, u'Пример написания
плагина для книги\n В.Петин Микрокомпьютеры Raspberry Pi. Практическое
руководство')
        self.addControl(self.label0)
        self.label1 = xbmcgui.ControlLabel(80, 110, 400, 50, u'Список ближайших
датчиков:')
        self.addControl(self.label1)
        self.label2 = xbmcgui.ControlLabel(600, 110, 300, 50, u'Показания:')
        self.addControl(self.label2)
        self.label3 = xbmcgui.ControlLabel(600, 150, 300, 50, u'Датчик:')
        self.addControl(self.label3)

    def getDevices(self):
        self.id_devices=[]
        data1 = {'cmd': 'sensorNear','uuid': md5_app_id,'api_key':
api_key,'radius': 50,'lat': 44.05,         'lng': 42.80,'lang': 'ru'}
        request = urllib2.Request('http://narodmon.ru/client.php',
json.dumps(data1))
        response = urllib2.urlopen(request)
```

```
        result = json.loads(response.read())
        i=0
        self.list = xbmcgui.ControlList(100, 150, 400, 500)
        self.addControl(self.list)
        for dev in result['devices']:
          el_address=dev['location'].split(", ")
          coords="( "+str(dev['lat'])+" , "+str(dev['lng'])+" )"
          title=el_address[0]+" - "+dev['name']+coords
          i=i+1
          listitem=xbmcgui.ListItem(title,str(i))
          self.list.addItem(listitem)
          #self.list.addItem(title)
          self.id_devices.append(dev['id'])
          #addDevice(title, dev['id'], 10)
        self.setFocus(self.list)
        self.label4 = xbmcgui.ControlLabel(600, 200, 500, 500, u'Данные:')
        self.addControl(self.label4)

    def getSensors(self,id_dev):
        data2 = {'cmd': 'sensorDev','uuid': md5_app_id,'api_key': api_key,'id':
1591,'lang': 'ru'}
        data2['id']=self.id_devices[id_dev-1]
        request = urllib2.Request('http://narodmon.ru/client.php',
json.dumps(data2))
        response = urllib2.urlopen(request)
        #  JSON
        value1=""
        result = json.loads(response.read())
        #self.list2 = xbmcgui.ControlList(600, 200, 400, 500)
        #self.addControl(self.list2)
        for sens in result['sensors']:
          if sens['type']==1:
             value1=value1+"\nТемпература "+str(sens['value'])+" C"
          elif sens['type']==2:
             value1=value1+"\nВлажность "+str(sens['value'])+" %"
          elif sens['type']==3:
             value1=value1+"\nДавление "+str(sens['value'])+" мм.рт.ст"
          else:
             value1=value1+"\n??????? "+sens['value']
          #self.list2.addItem(value1)
        self.label4.setLabel(value1)

    def onAction(self, action):
        # Если нажали ESC или Backspace...
        if action == ACTION_NAV_BACK or action == ACTION_PREVIOUS_MENU:
            # ...закрываем плагин.
            self.close()
```

```
    def onControl(self, control):
      if control == self.list:
        item = self.list.getSelectedItem()
        self.label3.setLabel(item.getLabel())
        self.getSensors(int(item.getLabel2()))
        #self.message('You selected : ' + item.getLabel2())
    def message(self, message):
      dialog = xbmcgui.Dialog()
      dialog.ok(" My message title", message)

if __name__ == '__main__':
    # Создаем экземпляр класса-контейнера.
    addon = MyAddon()
    # Выводим контейнер на экран.
    addon.doModal()
    # По завершении удаляем экземпляр.
    del addon
```

Рис. 5.50. Вывод списка устройств

Листинг 6.7. Файл addon.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<addon id="script.narodmonnaKMV"
   name="Narodmon"
   version="0.0.1"
   provider-name="Petin_V">
  <requires>
    <import addon="xbmc.python" version="2.0"/>
  </requires>
```

```
  <extension point="xbmc.python.script" library="default.py">
    <provides>executable</provides>
  </extension>
  <extension point="xbmc.addon.metadata">
    <platform>all</platform>
    <summary lang="en">Narodmon</summary>
    <description lang="en">Narodmon</description>
  </extension>
</addon>
```

Примечание

Файлы *default.py* и *addon.xml* (см. листинги 6.6 и 6.7), а также файл *icon.png* вы найдете в папке *glava_05\script.narodmonnaKMV* сопровождающего книгу электронного архива (см. *приложение*).

Для установки плагина заходим в меню системы Raspbmc **Система | Дополнения | Установить из файла ZIP** и указываем путь к архиву с нашим скриптом. Все, плагин установлен (рис. 5.51).

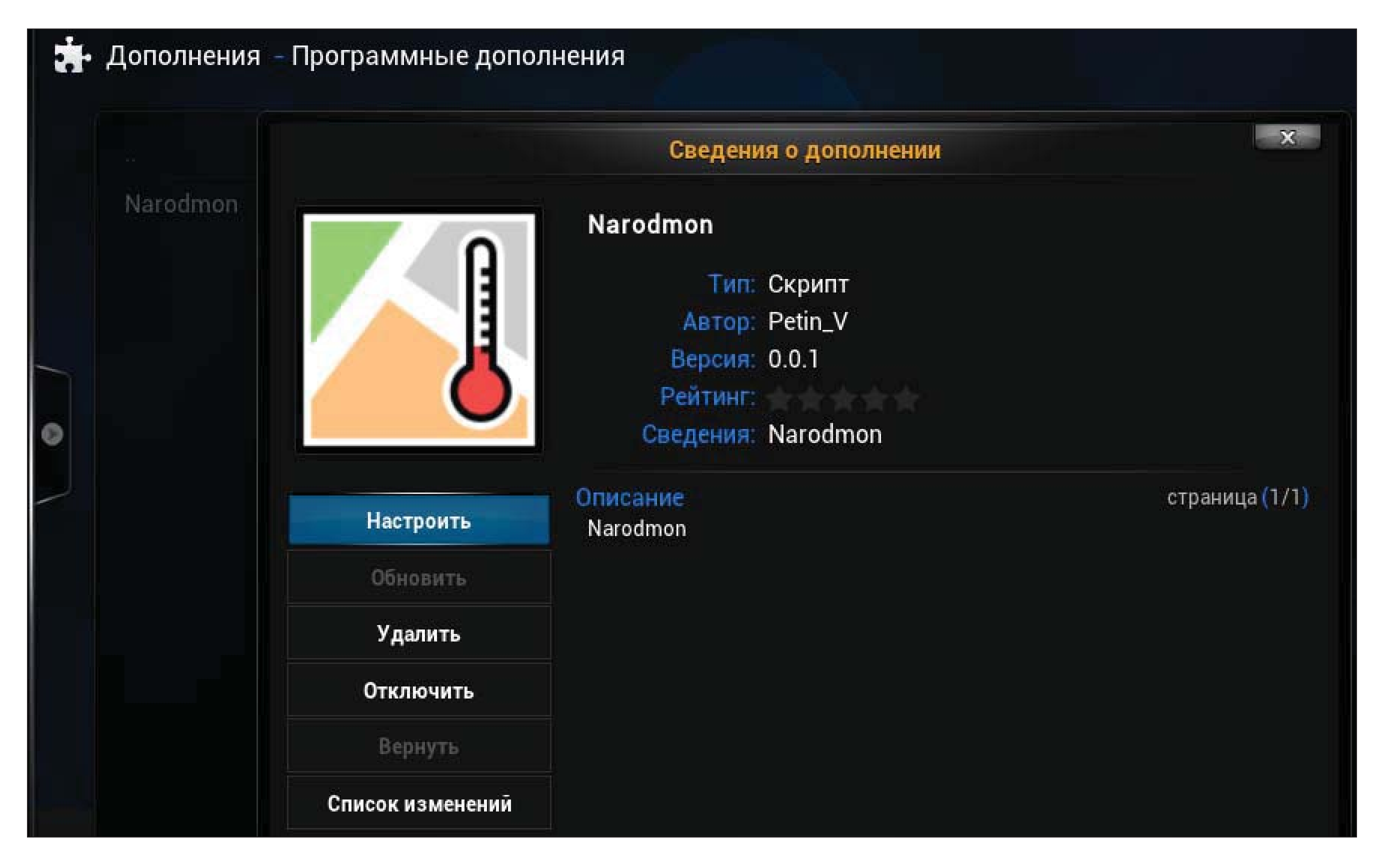


Рис. 5.51. Наш плагин в XBMC

Можно добавить нашему плагину универсальности, создав файл настроек settings.xml, позволяющий задавать координаты нужного места и радиус нахождения ближайших датчиков. Тогда в меню **Настройки** можно будет изменять эти переменные (рис. 5.52).

Содержимое файла settings.xml представлено в листинге 6.8.

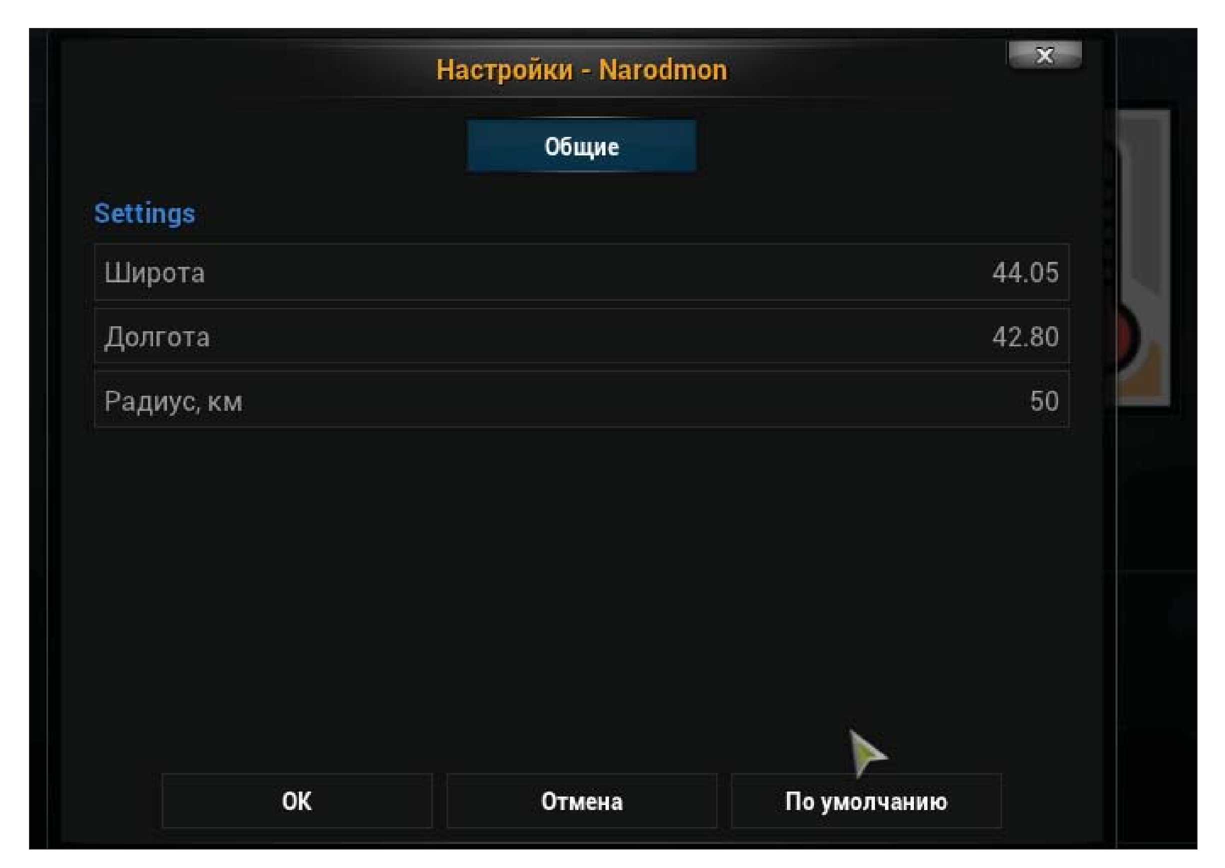


Рис. 5.52. Установка настроек

Листинг 6.8. Файл settings.xml

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<settings>
    <setting type="lsep" label="Settings" />
    <setting id="lat" type="text"  label="Широта" default="44.05"/>
    <setting id="lng" type="text"  label="Долгота" default="42.80"/>
    <setting id="radius" type="text"  label="Радиус, км" default="50"/>
</settings>
```

Примечание

Файл *settings.xml* (листинг 6.8) вы найдете в папке *glava_05\script.narodmonnaKMV\resources* сопровождающего книгу электронного архива (см. *приложение*).

Думаю, изменить плагин для внесения новой функциональности не составит для вас труда.



ГЛАВА 6

Работа с интерфейсом GPIO

Raspberry Pi несет на борту интерфейс, называемый GPIO (рис. 6.1), — набор портов ввода/вывода общего назначения (General Purpose Input/Output). Из 26 линий этого интерфейса для управления доступны только 17. На них реализованы интерфейсы UART, консольный порт, SPI (Serial Peripheral Interface, последовательный периферийный интерфейс) и I^{2}C (Inter-Integrated Circuit, последовательная шина данных для связи интегральных схем). Расстояние между выводами 2,54 мм. Выводы UART, SPI и I^{2}C в случае необходимости могут быть настроены как обычные порты ввода/вывода.

Назначение выводов GPIO (так называемая *распиновка*) представлено на рис. 6.2 и 6.3. В первой ревизии платы выводы 4, 9, 14, 17, 20, 25 обозначены как DNC (Do Not Connect), и подсоединять к ним что-либо не следует — плата может сгореть. На новых ревизиях платы разведены, но не распаяны, еще четыре GPIO, дающие дополнительные выводы I^{2}C и I^{2}S (Inter-Integrated Circuit, последовательная шина данных для связи интегральных схем). Использование GPIO — это как раз самое интересное и творческое применение Raspberry Pi.



Рис. 6.1. Порты GPIO на плате Raspberry Pi

С помощью выводов GPIO Raspberry Pi может управлять внешними устройствами. Устройства могут быть абсолютно любыми, область применения Raspberry Pi в этом плане ограничена лишь вашей фантазией и знаниями. Следующие разделы главы посвящены примерам использования GPIO, начиная с самых простых и заканчивая вариантами с применением плат расширения.

3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Рис. 6.2. Распиновка портов GPIO Revision 1

3.3V	1	2	5V
I2C1 SDA	3	4	5V
I2C1 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
GROUND		10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 27	13	14	GROUND
GPIO 22	15	16	GPIO 23
3.3V	17	18	GPIO 24
SP10 MOSI	19	20	GROUND
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
GROUND	25	26	SP10 CE1 N

Рис. 6.3. Распиновка портов GPIO Revision 2

6.1. Особенности работы с GPIO

При работе с портами GPIO следует помнить о некоторых их особенностях и соблюдать определенные меры предосторожности, чтобы не повредить Raspberry Pi. Вот основные из них:

- максимальный суммарный ток обоих выводов 3,3 В равен 50 мА, и эти выводы могут использоваться для питания внешних устройств только в том случае, если их потребляемый ток меньше 50 мА;
- максимальный суммарный ток обоих выводов 5 В равен 300 мА, и эти выводы также могут использоваться для питания внешних устройств только в том случае, если их потребляемый ток меньше 300 мА;
- на GPIO нельзя подавать напряжение больше 3,3 В! Цифровые выводы GPIO имеют уровни напряжения 0–3,3 В и не совместимы с традиционными уровнями напряжения 0–5 В. Если подать на вывод GPIO логическую единицу, представляющую собой 5 В (а не 3,3 В), — этот вывод может выйти из строя;
- выводы GPIO14 и GPIO15 по умолчанию выполняют альтернативную функцию и являются выводами UART (RXD и TXD), поэтому после включения на них

присутствует высокий уровень 3,3 В, однако программно их можно переконфигурировать в обычные выводы. Все остальные выводы GPIO после включения Raspberry Pi выполняют основную функцию и работают как обычные цифровые;

- все настраиваемые выводы GPIO — кроме GPIO0 (SDA) и GPIO1 (SCL) — по умолчанию являются входами, и поэтому имеют высокое входное сопротивление, при этом подтяжка логического уровня у них не включена, так что после включения Raspberry Pi напряжение на них может "плавать";
- выводы GPIO0 (SDA) и GPIO1 (SCL) по умолчанию "подтянуты" к питанию, поэтому после включения Raspberry Pi на них присутствует напряжение логической единицы (3,3 В);
- сигнал на любом из цифровых выводов может служить источником внешнего прерывания.

Нужно помнить, что GPIO — это выводы, непосредственно подключенные к процессору Raspberry Pi, они являются инструментом для взаимодействия с ним. Поэтому неосторожное обращение с GPIO может привести к необратимым последствиям для процессора.

Работать с GPIO можно двумя способами:

- используя оболочку bash и файловую систему Raspbian;
- используя языки программирования.

6.1.1. Управление GPIO из оболочки bash

ОС Raspbian представляет собой один из дистрибутивов Linux, а концепция Linux предполагает, что любой объект является файлом. Именно это позволяет выводить и считывать сигналы с GPIO обычными командами оболочки bash прямо в терминале. Вывод логической единицы при этом выглядит как команда записи "1" в файл, соответствующий нужному выводу:

```
sudo su -
echo "25" > /sys/class/gpio/export
echo "25" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio25/direction
echo "1" > /sys/class/gpio/gpio25/value
echo "0" > /sys/class/gpio/gpio25/value
```

Для чтения входов надо использовать команду `cat` и путь к файлу:

```
echo "24" > /sys/class/gpio/export
echo "in" > /sys/class/gpio/gpio0/direction
cat /sys/class/gpio/gpio24/value
```

Все операции должны выполняться от имени пользователя root.

6.1.2. Управление GPIO из языка Python

Для работы с GPIO на языке Python требуется специальная библиотека RPi.GPIO. В новом дистрибутиве Raspbian она уже установлена, а если у вас дистрибутив старый, то для установки библиотеки RPi.GPIO выполните команду:

```
sudo apt-get install python-rpi.gpio
```

Чтобы использовать эту библиотеку, необходимо в программу на Python добавить строку импорта библиотеки RPi.GPIO:

```
Import RPi.GPIO as GPIO
```

При подготовке программы можно выбрать один из двух способов нумерации портов GPIO. Первый — GPIO.BOARD — использует систему нумерации портов на плате Raspberry Pi. Преимущество этой системы нумерации в том, что ваше оборудование будет работать всегда, независимо от номера ревизии платы, — вам не придется перемонтировать свой разъем или изменять имеющийся код. Вторая система нумерации — GPIO.BCM (номера BCM). Это более низкий уровень работы — с прямым обращением к номерам каналов на процессоре (SoC) Broadcom. Выбор способа нумерации определяется соответствующими командами:

```
GPIO.setmode(GPIO.BOARD)
GPIO.setmode(GPIO.BCM)
```

Следующие команды устанавливают режим работы контакта на вход или выход:

```
GPIO.setup(channel, GPIO.IN)
GPIO.setup(channel, GPIO.OUT)
```

Если входной канал ни к чему не подключен, его значение может "плыть". Следующие команды устанавливают начальную "подтяжку" вывода к питанию или к "земле":

```
GPIO.setup(channel, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(channel, GPIO.IN, GPIO.PUD_DOWN)
```

Для выходов `OUT` можно установить начальное значение 0 или 1:

```
GPIO.setup(channel, GPIO.OUT, GPIO.LOW)
GPIO.setup(channel, GPIO.OUT, GPIO.HIGH)
```

Для чтения значения контакта GPIO, настроенного как вход `IN`, служит следующая команда:

```
GPIO.input(channel)
```

Значение контакта, настроенного как выход `OUT`, устанавливается следующей командой:

```
GPIO.output(channel, state)
```

В листинге 6.1 приведен пример использования команд работы с GPIO.

Листинг 6.1. Пример использования команд для работы с GPIO

```
import RPi.GPIO as GPIO        #подключаем библиотеку
     GPIO.setmode(GPIO.BCM)    #устанавливаем режим нумерации
     GPIO.setup(7, GPIO.OUT)   #конфигурируем GPIO 7 как выход
     GPIO.setup(8, GPIO.IN)    #конфигурируем GPIO 8 как вход
     GPIO.output(7, True)      #выводим на GPIO 7 логическую "1" (3.3 V)
     GPIO.output(7, False)     #выводим на GPIO 7 логический "0"
     signal = GPIO.input(8)    #считываем с GPIO 8 в переменную signal
     GPIO.cleanup()            #завершаем работу с GPIO
```

Библиотека RPi.GPIO позволяет использовать контакты GPIO в качестве выходов ШИМ (сигналов широтно-импульсной модуляции).

Для создания экземпляра ШИМ служит команда:

```
p = GPIO.PWM(channel, frequency)
```

где `frequency` — частота, Гц.

Для старта ШИМ на контакте:

```
p.start(dc)
```

где `dc` — рабочий цикл ШИМ (0,0–100,0).

Для изменения частоты сигнала:

```
p.ChangeFrequency(freq)
```

где `freq` — частота сигнала, Гц.

Для изменения рабочего цикла ШИМ:

```
p.ChangeDutyCycle(dc)  # where 0.0 <= dc <= 100.0
```

Останов выдачи сигнала ШИМ на контакте:

```
p.stop()
```

В листинге 6.2 представлен пример плавного включения/выключения светодиода, подключенного к контакту.

Листинг 6.2. Пример плавного включения/выключения светодиода

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12, 50)  # контакт 12 частота 50 Гц
p.start(0)
try:
    while 1:
```

```
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```

Функция ожидания события — изменения состояния на входе `IN` — выглядит следующим образом:

```
GPIO.wait_for_edge(channel, GPIO.RISING)
GPIO.wait_for_edge(channel, GPIO.FALLING)
GPIO.wait_for_edge(channel, GPIO.BOTH)
```

Эта функция прерывает выполнение программы до изменения состояния на входе GPIO. В отличие от нее функция `add_event_detected()` выполняется в цикле программы, и для того, чтобы узнать об изменении состояния на контакте GPIO, необходимо в цикле проверять наступление события `event_detected()`:

```
GPIO.add_event_detect(channel, GPIO.RISING
................
if GPIO.event_detected(channel):
    print('Button pressed')
```

Каждый контакт GPIO может быть настроен на работу в режиме прерывания, в этом случае при наступлении события на контакте управление передается функции обработки прерывания. Функция обработки прерывания работает в отдельном потоке, не прерывая выполнения основной программы:

```
def my_callback(channel):
    print('обработка прерывания!')
GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback)
```

В конце любой программы рекомендуется очистить все ресурсы, которые могли использоваться. Для такой очистки в конце скрипта надо предусмотреть команду:

```
GPIO.cleanup()
```

Для сброса одного контакта служит команда:

```
GPIO.cleanup(channel)
```

6.1.3. Управление GPIO из языка С

Для работы с GPIO на языке С требуется специальная библиотека bcm2835, скачать которую можно с сайта **http://www.open.com.au/mikem/bcm2835/index.html**. В помощь тем, кто привык писать программы для Arduino, Гордон Хендерсон (Gordon

Henderson, **https://projects.drogon.net/**) написал Arduino-подобную библиотеку на C (WiringPi), которую мы и установим:

```
cd /tmp
wget http://project-downloads.drogon.net/files/wiringPi-1.tgz
tar xfz wiringPi-1.tgz
cd wiringPi/wiringPi
make
sudo make install
```

Для примера напишем с использованием библиотеки `wiringPi` программу мигания светодиодом, подключенным к выводу GPIO4, с периодичностью 2 секунды. Светодиод подключаем к контакту GPIO4(7) и через резистор 220 Ом на "землю". Текст этой программы (файл blink.c) представлен в листинге 6.3.

Листинг 6.3. Файл blink.c

```
#include <wiringPi.h>
#include <stdio.h>

int main (void)
{
  int pin = 7;
  printf("Raspberry Pi wiringPi blink test\n");
  if (wiringPiSetup() == -1)
    exit (1);

  pinMode(pin, OUTPUT);
  for (;;){
    printf("LED On\n");
    digitalWrite(pin, 1);
    delay(1000);
    printf("LED Off\n");
    digitalWrite(pin, 0);
    delay(1000);
  }
  return 0;
}
```

Компилируем код:

```
gcc -o blink blink.c -L/usr/local/lib -lwiringPi
```

И запускаем:

```
sudo ./blink
```

Светодиод должен мигать.

6.1.4. Подключение к Raspberry Pi жидкокристаллического дисплея

Рассмотрим в качестве примера подключение к Raspberry Pi через выводы GPIO жидкокристаллического (ЖК) дисплея wh1602 на базе контроллера HD44780. Этот монохромный дисплей (рис. 6.4) имеет опциональную подсветку и может отображать 2 строки по 16 символов. Разрешение символов — 5×8 точек.

Схема подключения ЖК-дисплея к выводам GPIO представлена на рис. 6.5.

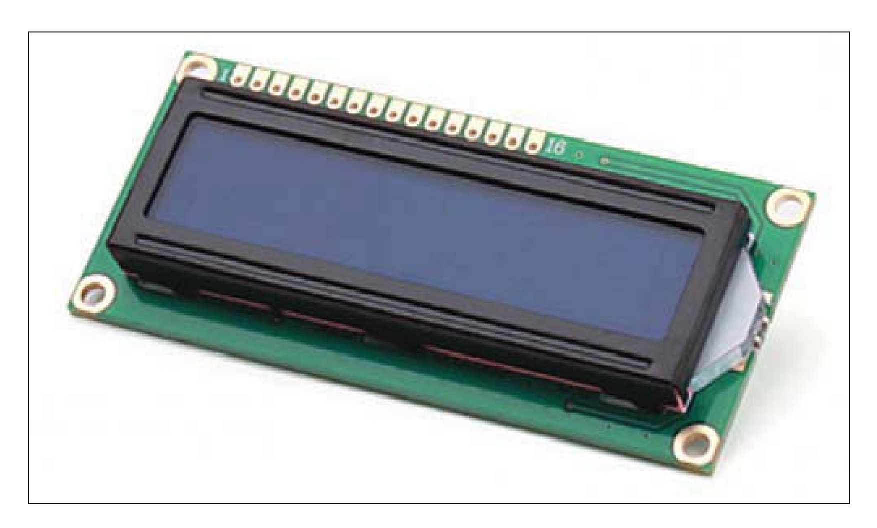


Рис. 6.4. Монохромный ЖК-дисплей wh1602

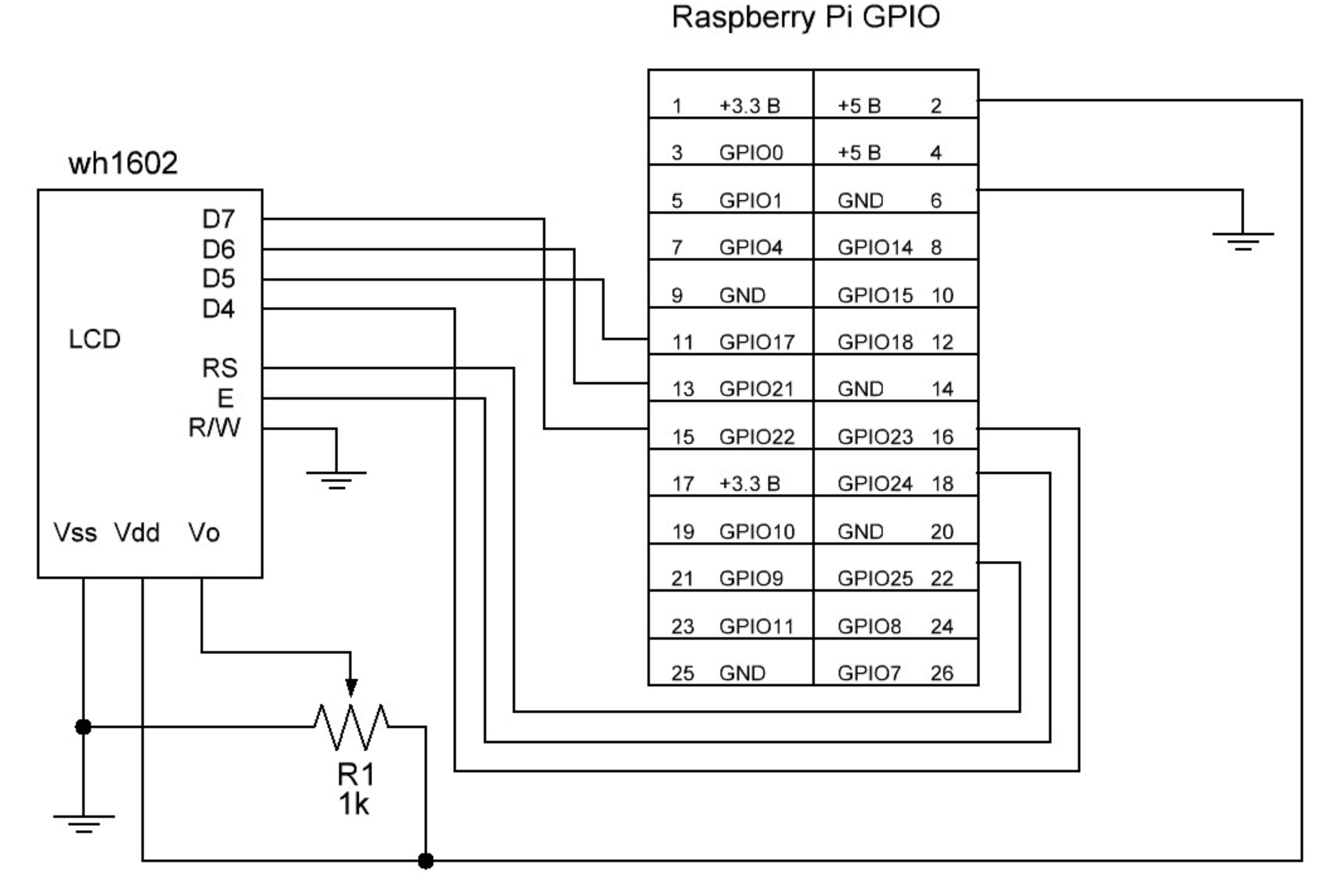


Рис. 6.5. Схема подключения дисплея к выводам GPIO

ЖК-дисплей wh1602 имеет режим самотестирования, который можно включить, подсоединив выводы:

- 1 — Vss;
- 2 — Vdd;
- 3 — Vo.

Верхний ряд символов дисплея должен целиком заполниться темными прямоугольниками (рис. 6.6).

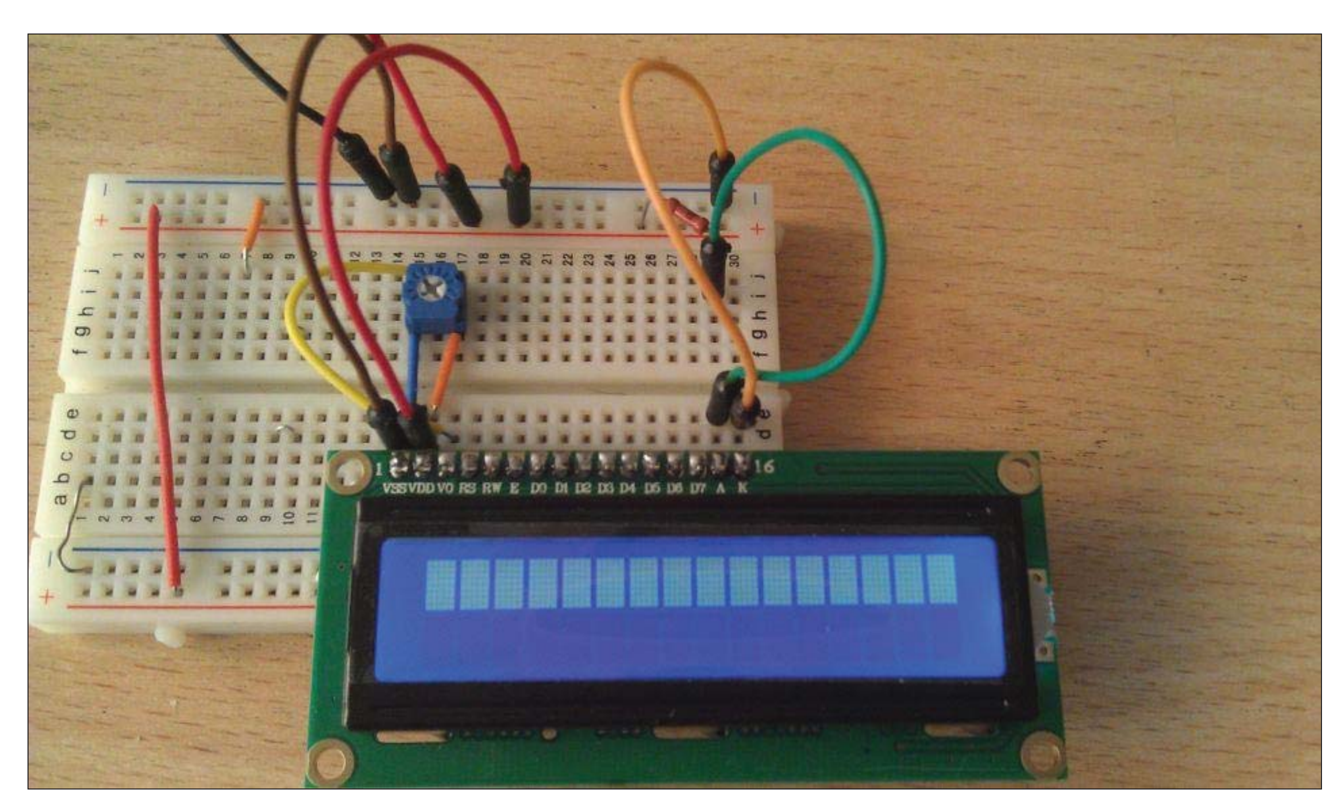


Рис. 6.6. Тест дисплея

Дисплей wh1602, как и прочие на контроллере HD44780, поддерживает два варианта параллельного интерфейса:

- 8-битный, выводы DB0–DB7, за один такт передается 1 байт (8 битов);
- 4-битный, выводы DB4–DB7, за один такт передается половина байта (4 бита).

Класс Python для управления жидкокристаллическим дисплеем доступен на сайте Github (**https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code**):

```
git clone git://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code.git
cd Adafruit-Raspberry-Pi-Python-Code
cd Adafruit_CharLCD
```

В листинге 6.4 приведено содержимое скрипта (файл LCD_example.py) для вывода на дисплей информации о загрузке и температуре процессора. Скрипт основан на классе языка Python `Adafruit_CharLCD` для управления дисплеем.

Листинг 6.4. Файл LCD_example.py.py

```
#!/usr/bin/python

from Adafruit_CharLCD import Adafruit_CharLCD
from subprocess import *
from time import sleep, strftime
from datetime import datetime

lcd = Adafruit_CharLCD()

cmd1 = "cat /proc/loadavg"
cmd2 = "/opt/vc/bin/vcgencmd measure_temp"

lcd.begin(16,2)

def run_cmd(cmd):
        p = Popen(cmd, shell=True, stdout=PIPE)
        output = p.communicate()[0]
        return output

lcd.clear()
lcd.message("  Adafruit 16x2\n  Standard LCD")
sleep(3)

while 1:
       lcd.clear()
       load = run_cmd(cmd1)
       temp = run_cmd(cmd2)
       #lcd.message(datetime.now().strftime('%b %d  %H:%M:%S\n'))
       #lcd.message('IP %s' % ( ipaddr ) )
       loads=load.split(' ')
       loadavg=int(float(loads[0])*100)
       lcd.message("loadavg="+str(loadavg)+"%\n" )
       lcd.message(temp )
       sleep(2)

GPIO.cleanup()
```

На рис. 6.8 показан вывод результатов работы скрипта на экран дисплея.

Примечание

Код этого скрипта (файл *LCD_example.py*) и код класса для управления дисплеем (файл *Adafruit_CharLCD.py*) вы найдете в папке *glava_06\lcd1602* сопровождающего книгу электронного архива (см. *приложение*).

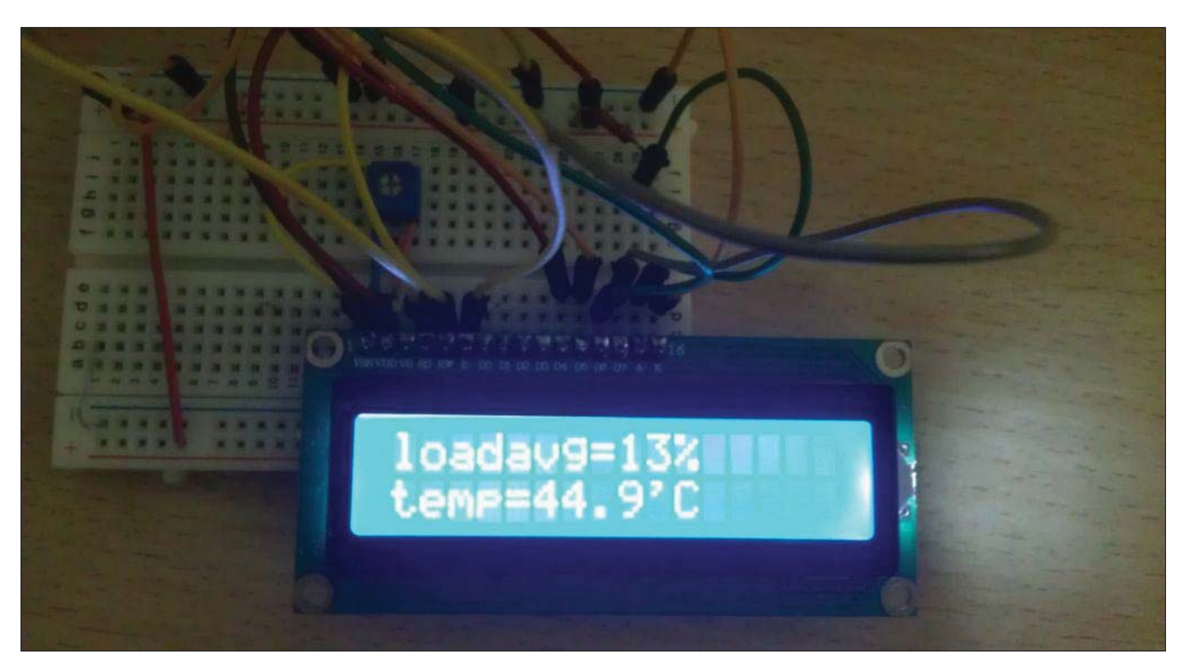


Рис. 6.7. Результат работы скрипта

6.1.5. Схема подключения телевизионного пульта к Raspberry Pi на дистрибутиве Raspbmc

Рассмотрим процесс подключения телевизионного пульта к мини-ПК Raspberry Pi на дистрибутиве Raspbmc. В этом примере использован инфракрасный (ИК) ресивер SM3374 (рис. 6.8). Схема подключения ресивера представлена на рис. 6.9.

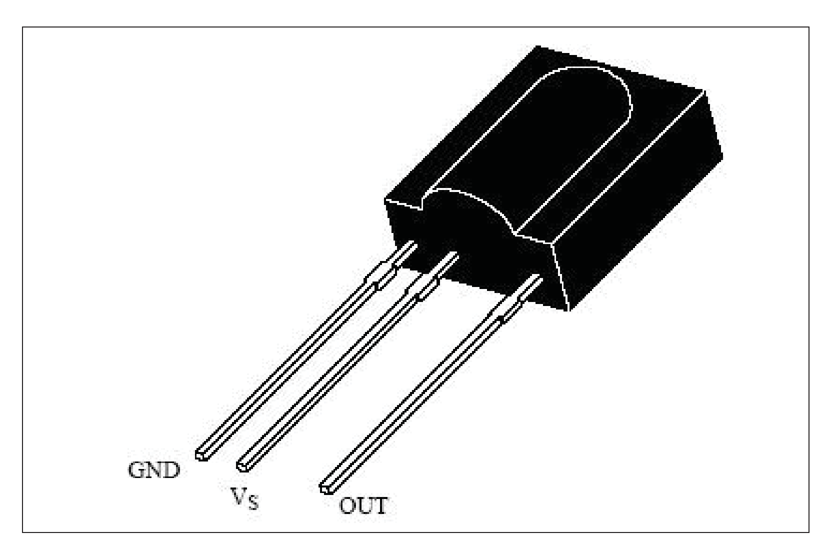


Рис. 6.8. ИК-ресивер SM3374

Декодировать инфракрасные сигналы многих (но не всех) обычно используемых пультов дистанционного управления позволяет пакет LIRC (**L**inux **I**nfrared **R**emote Control, инфракрасный ПДУ для Linux). LIRC запускается как демон, который декодирует ИК-сигналы, полученные от драйверов устройств, и предоставляет на сокете соответствующую информацию. Нам нужно установить этот пакет и его клиентские библиотеки. Все действия производим через SSH-соединение:

```
sudo apt-get install lirc liblircclient-dev
```

Вносим изменения в файл /etc/modules, записав туда строки:

```
lirc_dev
lirc_rpi gpio_in_pin=18
```

Это приведет к запуску модуля при загрузке.

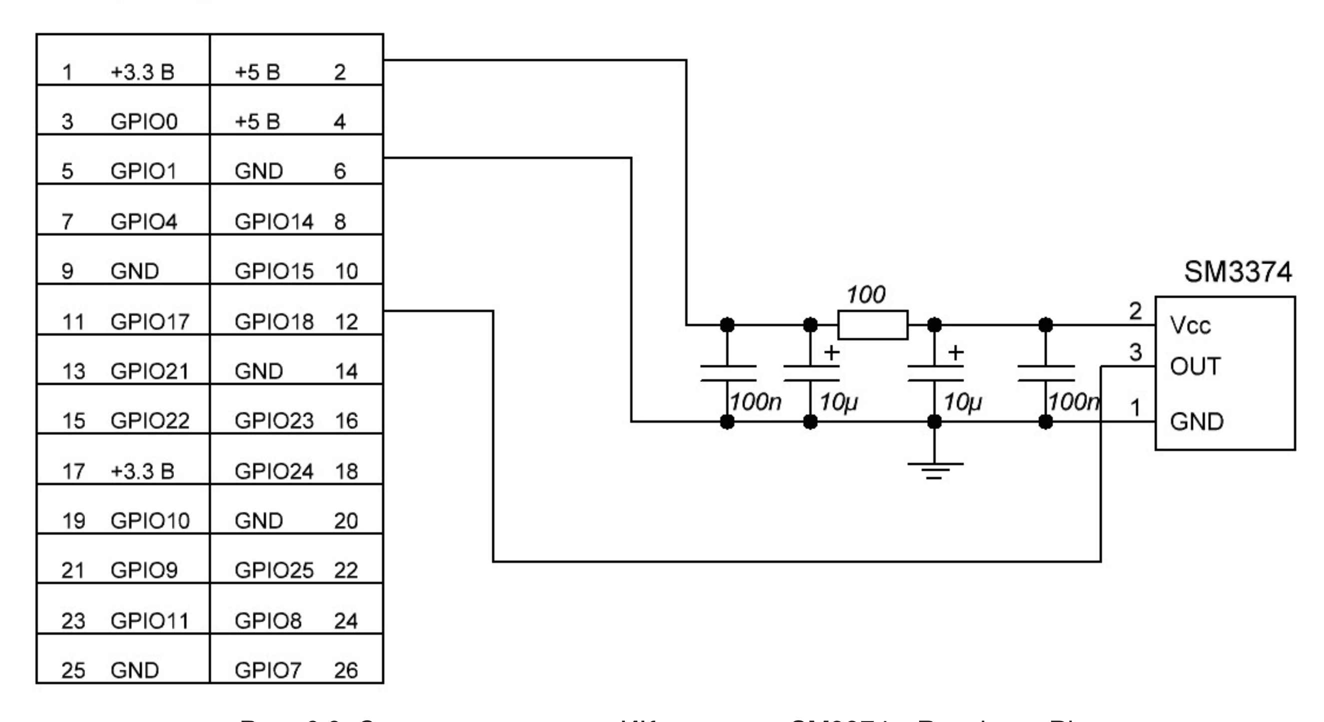


Рис. 6.9. Схема подключения ИК-ресивера SM3374 к Raspberry Pi

Изменяем содержимое файла /etc/lirc/hardware.conf (листинг 6.5).

Листинг 6.5. Файл /etc/lirc/hardware.conf

```
# /etc/lirc/hardware.conf
#
# Arguments which will be used when launching lircd
LIRCD_ARGS="--uinput"

#Don't start lircmd even if there seems to be a good config file
#START_LIRCMD=false

#Don't start irexec, even if a good config file seems to exist.
#START_IREXEC=false

#Try to load appropriate kernel modules
LOAD_MODULES=true

# Run "lircd --driver=help" for a list of supported drivers.
DRIVER="default"
```

```
# usually /dev/lirc0 is the correct setting for systems using udev
DEVICE="/dev/lirc0"
MODULES="lirc_rpi"

# Default configuration files for your hardware if any
LIRCD_CONF=""
LIRCMD_CONF=""
```

Для выполнения быстрого теста, чтобы увидеть, что LIRC работает, необходимо остановить демон `lirc` и запустить `mode2`:

```
sudo modprobe lirc_rpi
sudo kill $(pidof lircd)
mode2 -d /dev/lirc0
```

При нажатии кнопок на пульте дистанционного управления `mode2` выведет в терминал длины инфракрасных импульсов:

```
pulse 168
space 4816
pulse 169
space 4784
pulse 1333
space 3638
```

Теперь нам нужно раздобыть файл конфигурации lircd.conf с кодами кнопок. На сайте **http://lirc.sourceforge.net/remotes** можно найти уже готовые файлы конфигурации для очень многих пультов. Есть там такой файл и для используемого мной пульта LG 6710V00090D. Содержимое его представлено в листинге 6.6.

Листинг 6.6. Файл конфигурации lircd.conf

```
begin remote
  name  LG_6710V00090D
  bits          13
  flags RC5
  eps           30
  aeps         100

  one          805   876
  zero         805   876
  plead        713
  gap       121999
  toggle_bit     2


      begin codes
          power                    0x100C
          mute                     0x100D
```

```
          1                         0x1001
          2                         0x1002
          3                         0x1003
          4                         0x1004
          5                         0x1005
          6                         0x1006
          7                         0x1007
          8                         0x1008
          9                         0x1009
          0                         0x1000
          eye                       0x1037
          i-ii                      0x1036
          pr-up                     0x1020
          pr-down                   0x1021
          vol-down                  0x1011
          vol-up                    0x1010
          ok                        0x1025
          q-view                    0x1032
          list                      0x1034
          psm                       0x100E
          ssm                       0x1016
          sleep                     0x1026
          picture                   0x1013
          sound                     0x1024
      end codes
end remote
```

Закачанный файл конфигурации необходимо переименовать в lircd.conf и скопировать в папку etc/lirc. Если файл конфигурации вашего пульта найти не удастся, для его создания можно воспользоваться программой `irrecord`:

```
irrecord –d /dev/lirc0 ~/lircd.conf
```

Запустив программу, нажимаем кнопки на пульте до тех пор, пока на экране не появятся 80 точек, и вбиваем наименования кнопок. Затем опять нажимаем кнопки. Получившийся файл lircd.conf и поместим в папку /etc/lirc.

Теперь заходим на мини-ПК Raspberry Pi и через меню Raspbmc **Программы | Raspbms Settings | IR Remote** устанавливаем поддержку нашего пульта, указав конфигурационный файл lircd.conf (рис. 6.10).

Запускаем сервис:

```
service lirc start
```

Набираем:

```
irw
```

Теперь при нажатии на кнопки ИК-пульта LIRC передает системе названия кнопок.

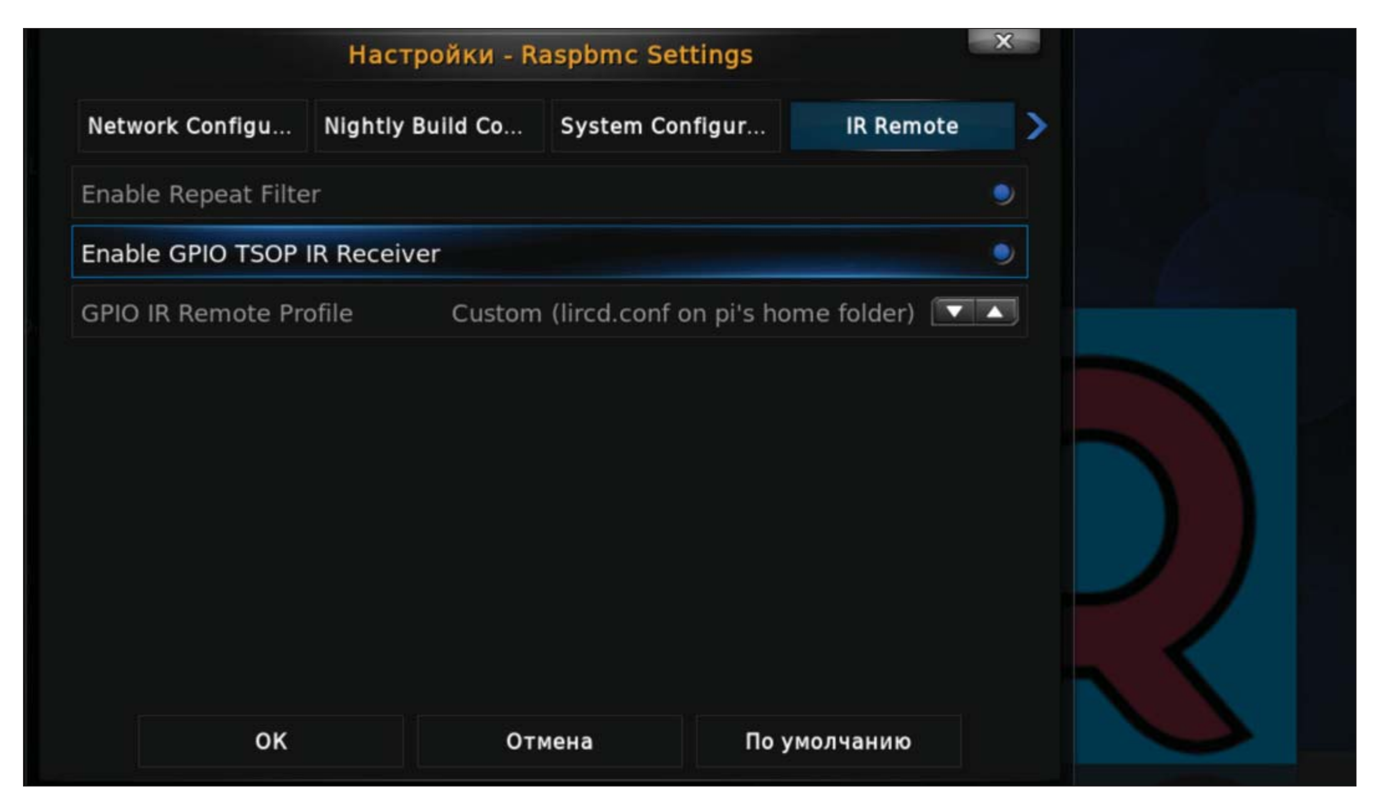


Рис. 6.10. Установка поддержки ИК-пульта в Raspbmc

В *разд. 5.12* мы ставили задачей использовать пульт в медиапроигрывателе Raspbmc. Для этого необходимо создать файл ~/.xbmc/userdata/Lircmap.xml, ввести в нужные поля название устройства, функции XBMC и соответствующие им названия кнопок пульта из файла lircd.conf. Пример файла Lircmap.xml приведен в листинге 6.7. В нем задействованы всего четыре кнопки.

Листинг 6.7. Файл Lircmap.xml

```
<lircmap>
<remote device=" LG_6710V00090D">
<left>vol-down</left>
<right>vol-up</right>
<up> pr-up </up>
<down> pr-down  </down>
<enter>ok</enter>
</remote>
</lircmap>
```

6.2. Доступ к портам GPIO через веб-интерфейс

Для доступа к портам GPIO через веб-интерфейс мы воспользуемся WebIOPi — фреймворком, позволяющим контролировать состояние и управлять всеми портами GPIO локально или удаленно, из браузера или любого приложения.

Возможности WebIOPi:

- REST API через HTTP и CoAP с поддержкой мультикаста;
- работа с GPIO, Serial, I^2C, SPI, 1-Wire;
- встроенная поддержка более чем 30 устройств, включая ЦАП, АЦП, датчики;
- совместимость с Python 2 и 3;
- защита логином/паролем;
- множество примеров.

Для установки фреймворка WebIOPi на ОС Raspbian необходимо скачать соответствующий архив, извлечь содержащиеся в нем файлы и запустить сценарий установки, который автоматически загрузит и инсталлирует необходимые зависимости:

```
wget http://webiopi.googlecode.com/files/WebIOPi-0.6.0.tar.gz
tar xvzf WebIOPi-0.6.0.tar.gz
cd WebIOPi-0.6.0
sudo ./setup.sh
```

После достаточно продолжительной установки WebIOPi готов к запуску. Лучше запускать его как сервис:

```
sudo /etc/init.d/webiopi start
```

Если нужно, чтобы WebIOPi стартовал автоматически при загрузке системы, выполните следующую команду:

```
sudo update-rc.d webiopi defaults
```



Рис. 6.11. Стартовая страница WebIOPi

Теперь можно открыть веб-браузер на любом компьютере в домашней сети и набрать адрес: `http://iprasp:8000`, где `iprasp` — IP-адрес Raspberry Pi. Имя пользователя: `webiopi`, пароль: `raspberry`. В результате в браузере откроется стартовая страница WebIOPi (рис. 6.11).

На странице **GPIO-Header** управления выводами GPIO (рис. 6.12) можно задать режим работы любой ножки и установить значение на выходе.

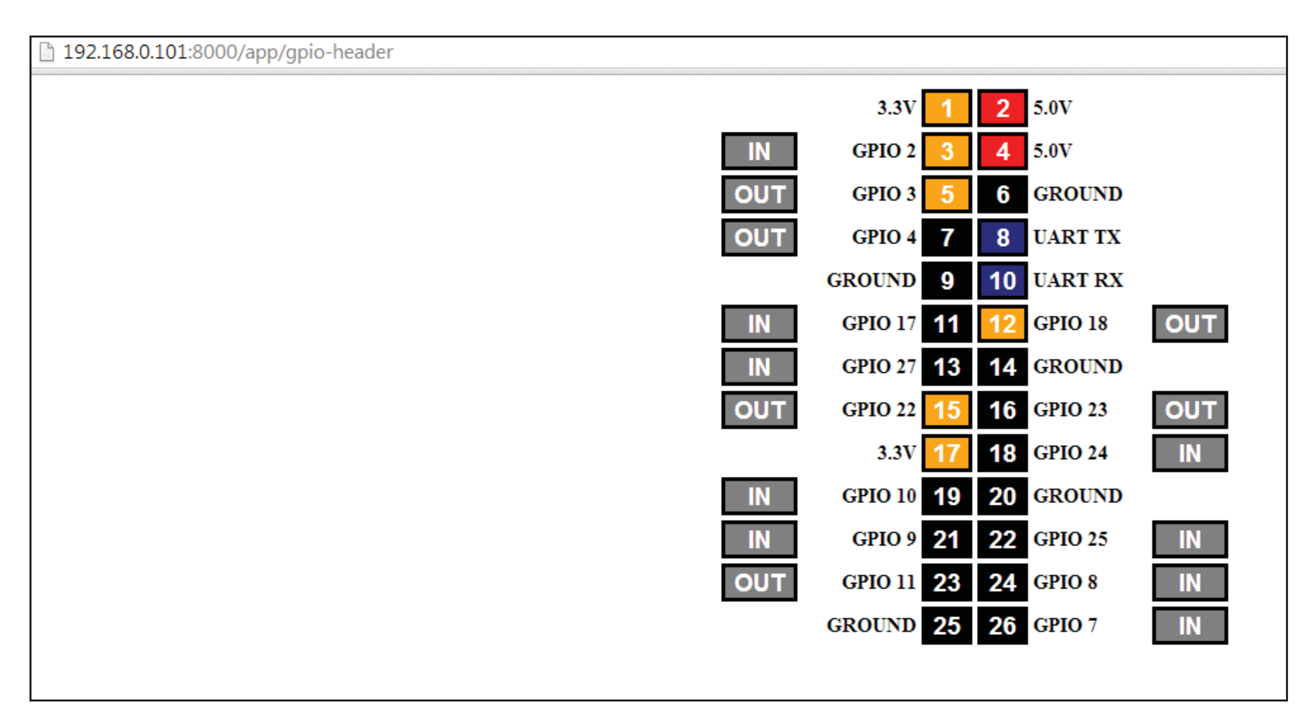


Рис. 6.12. Страница управления выводами GPIO

6.2.1. Установка пользовательского пароля WebIOPi

Сервер WebIOPi использует зашифрованный файл etc/webiopi/passwd, содержащий логин и пароль (как уже отмечалось ранее, по умолчанию логин: `webiopi`, пароль: `raspberry`). Чтобы изменить пароль, необходимо выполнить команду:

```
webiopi-passwd
```

и далее следовать инструкциям (рис. 6.13).

```
pi@raspberrypi: ~/WebIOPi-0.6.0
pi@raspberrypi ~/WebIOPi-0.6.0 $ sudo webiopi-passwd
WebIOPi passwd file generator
Enter Login: webiopi
Enter Password:
Confirm password:

Hash: 29cc67d7e1df8f2d56c215da9d474bfea6b74597322cce42dbb2fbcff8feccle
Saved to /etc/webiopi/passwd
pi@raspberrypi ~/WebIOPi-0.6.0 $
```

Рис. 6.13. Изменение пароля WebIOPi

После изменения пароля сервер необходимо перезагрузить:

```
sudo /etc/init.d/webiopi restart
```

Для снятия защиты при входе в WebIOPi необходимо либо ввести пустой пароль, либо удалить файл /etc/webiopi/passwd.

6.2.2. Настройка WebIOPi

Настраивается сервер WebIOPi внесением изменений в файл его конфигурации /etc/webiopi/config. Синтаксис этого файла такой же, как и у прочих INI-файлов, — он содержит несколько разделов, содержащих пары "ключ = значение".

Блок `[HTTP]` позволяет включить или отключить HTTP, а также изменить значение порта. В этом блоке можно изменить местоположение файла passwd, домашней папки и название индексного файла HTML:

```
[HTTP]
enabled = true
port = 8000
passwd-file = /etc/webiopi/passwd
doc-root = /home/pi/webiopi/examples/scripts/macros
welcome-file = index.html
```

Блок `[COAP]` позволяет включить или отключить сервер COAP, а также изменить значение порта:

```
[COAP]
enabled = true
port = 5683
multicast = true
```

Блок `[GPIO]` позволяет установить пользовательские настройки и значения для портов GPIO при запуске WebIOPi. Например:

```
[GPIO]
21 = IN
23 = OUT 0
24 = OUT 0
25 = OUT 1
```

Блок `[~GPIO]` позволяет установить пользовательские настройки и значения для портов GPIO при перезагрузке WebIOPi. Например:

```
[~GPIO]
21 = IN
23 = IN
24 = IN
25 = OUT 0
```

Блок `[SCRIPTS]` определяет список скриптов, выполняемых при запуске WebIOPi. Например:

```
[SCRIPTS]
myscript = /home/pi/webiopi/examples/scripts/macros/script.py
```

Блок `[REST]` позволяет с помощью REST API (см. *разд. 6.2.3*) ограничить доступ `GET`/`POST` к некоторым портам:

```
 [REST]
gpio-export = 21, 23, 24, 25
gpio-post-value = false
gpio-post-function = false
device-mapping = false
```

Блок `[DEVICES]` позволяет подключить устройства, поддерживаемые WebIOPi, к конкретным портам GPIO (список устройств, поддерживаемых WebIOPi, мы рассмотрим в *разд. 6.2.6*):

```
usb0 = Serial device:ttyUSB0 baudrate:9600
adc = MCP3008
dac = MCP4922 chip:1
gpio0 = MCP23017
gpio1 = MCP23017 slave:0x21
gpio2 = MCP23017 slave:0x22
pwm0 = PCA9685
pwm1 = PCA9685 slave:0x41
```

Блок `[ROUTES]` определяет список маршрутов переадресации. Это позволяет при использовании REST API скрыть в адресной строке значение порта доступа или другое назначение, т. е. придать адресам удобный вид:

```
/bedroom/light = /GPIO/25/value
/bedroom/temperature = /devices/temp2/sensor/temperature/c
```

6.2.3. Библиотека Javascript

WebIOPi включает в себя HTTP-сервер, обеспечивающий как HTML-ресурсы, так и интерфейс REST API для управления выводами GPIO. При запуске сервера браузер сначала загружает HTML-файл с включенной Javascript-библиотекой webiopi.js, включающей библиотеку jQuery для асинхронных вызовов к REST API. Этот метод очень эффективен, потому что не требует для обновления данных обновления страницы. Расширить возможности WebIOPi можно путем загрузки созданного с использованием Arduino-подобного синтаксиса пользовательского сценария Python, содержащего функции настройки выводов GPIO (рис. 6.14).

Для подключения библиотеки webiopi.js на странице HTML в блоке `[HEADER]` помещаем следующий код:

```
<script type="text/javascript" src="/webiopi.js"></script>
```

Функции библиотеки webiopi.js

Пояснение

Здесь и далее `webiopi()` — возвращаемый объект WebIOPi.

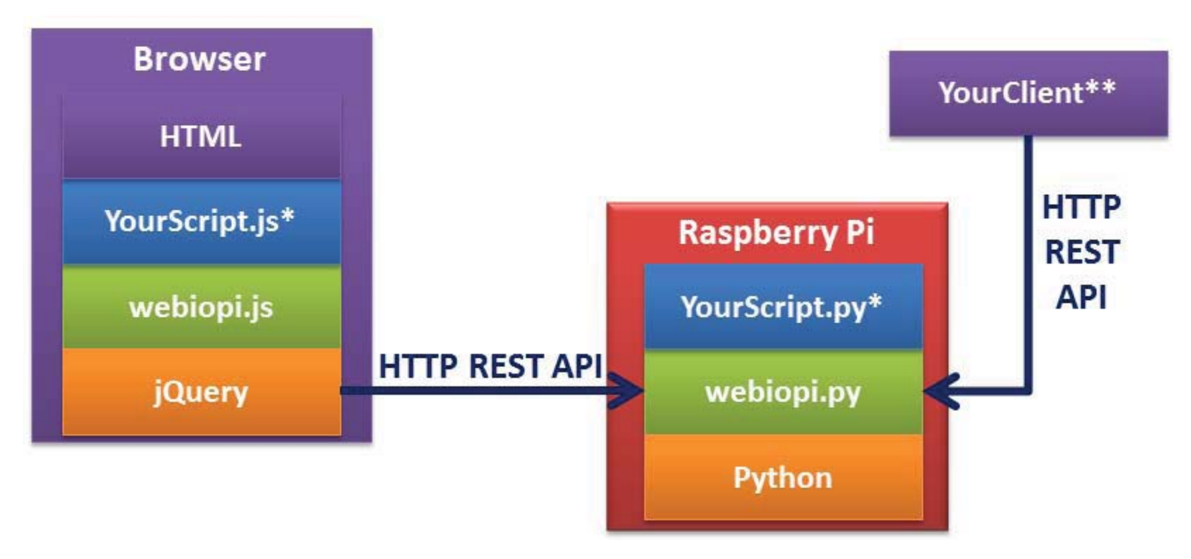


Рис. 6.14. Схема WebIOPi

Функция *WebIOPi.ready*

Функция `WebIOPi.ready()` регистрирует функцию обратного вызова при загрузке библиотеки webiopi.js.

Синтаксис: `WebIOPi.ready(callback)`

Параметр: `callback` — функция обратного вызова.

Функция *WebIOPi.setFunction*

Функция `WebIOPi.setFunction()` устанавливает назначение вывода GPIO.

Синтаксис:

- `WebIOPi.setFunction(gpio, func)`
- `WebIOPi.setFunction(gpio, func, callback)`

Параметры:

- `gpio` — номер вывода GPIO;
- `func` — назначение вывода:
  - `IN` — вывод конфигурирован как вход;
  - `OUT` — вывод конфигурирован как выход;
  - `PWM` — вывод конфигурирован как ШИМ-выход.
- `callback` — функция обратного вызова.

Функция *WebIOPi.digitalWrite*

Функция `WebIOPi.digitalWrite()` устанавливает цифровое значение вывода GPIO, сконфигурированного как выход (`OUT`).

Синтаксис:

- `WebIOPi.digitalWrite(gpio, value)`
- `WebIOPi.digitalWrite(gpio, value, callback)`

Параметры:

- `gpio` — номер вывода GPIO;
- `value` — значение для вывода (0 или 1);
- `callback` — функция обратного вызова.

Функция *WebIOPi.digitalRead*

Функция `WebIOPi.digitalRead()` получает значение с вывода GPIO, сконфигурированного как вход (`IN`).

Синтаксис:

- `WebIOPi.digitalRead(gpio)`
- `WebIOPi.digitalRead(gpio, callback)`

Параметры:

- `gpio` — номер вывода GPIO;
- `callback` — функция обратного вызова.

Возвращаемое значение: `value` (0 или 1).

Функция *WebIOPi.toggleValue*

Функция `WebIOPi.toggleValue()` переключает значение на выводе GPIO на противоположное.

Синтаксис: `WebIOPi. toggleValue(gpio)`

Параметр: `gpio` — номер вывода GPIO.

Функция *WebIOPi.callMacro*

Функция `WebIOPi.callMacro()` выполняет макрофункцию на сервере WebIOPi. Макрос может объявляться в скрипте Python, запускаемом при старте WebIOPi.

Синтаксис:

- `WebIOPi.callMacro(macro)`
- `WebIOPi.callMacro(macro, args)`
- `WebIOPi.callMacro(macro, args, callback)`

Параметры:

- `macro` — наименование макрофункции;
- `args` — массив передаваемых аргументов для макрофункции;
- `callback` — функция обратного вызова.

Функция *WebIOPi.outputSequence*

Функция `WebIOPi.outputSequence()` отправляет последовательность битов на выход GPIO.

Синтаксис:

- `WebIOPi.outputSequence(gpio, period, sequence)`
- `WebIOPi.outputSequence(gpio, period, sequence, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `period` — время отправки мсек одного бита;
- `sequence` — последовательность битов;
- `callback` — функция обратного вызова.

Пример:

```
var sequence = "01010100110011001100101010";
// отправка последовательности на gpio 7 с периодом 100 мсек
webiopi().outputSequence(7, 100, sequence, sequenceCallback);
```

Функция *WebIOPi.pulse*

Функция `WebIOPi.pulse()` отправляет импульс на выход GPIO.

Синтаксис:

- `WebIOPi.pulse(gpio)`
- `WebIOPi.pulse(gpio, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `callback` — функция обратного вызова.

Функция *WebIOPi.pulseRatio*

Функция `WebIOPi.pulseRatio()` отправляет ШИМ-сигналы на выход GPIO.

Синтаксис:

- `WebIOPi.pulseRatio (gpio, ratio)`
- `WebIOPi.pulseRatio (gpio, ratio, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `ratio` — значение для сигнала ШИМ (0,0–1,0);
- `callback` — функция обратного вызова.

Функция *WebIOPi.pulseAngle*

Функция `WebIOPi.pulseAngle()` отправляет ШИМ-сигналы на выход GPIO. Эта функция удобна при управлении сервоприводами для установки угла поворота рабочего органа от –45 до +45 градусов.

Синтаксис:

- `WebIOPi.pulseAngle (gpio, angle)`
- `WebIOPi.pulseAngle(gpio, angle, callback)`

Параметры:

- `gpio` — вывод GPIO;
- `angle` — значение сигнала ШИМ (от –45 до +45);
- `callback` — функция обратного вызова.

Функция *WebIOPi.createButton*

Функция `WebIOPi.createButton()` возвращает объект — простую кнопку.

Синтаксис:

- `WebIOPi.createButton(id, label)`
- `WebIOPi.createButton(id, label, mousedown)`
- `WebIOPi.createButton(id, label, mousedown, mouseup)`

Параметры:

- `id` — идентификатор кнопки;
- `label` — надпись на кнопке;
- `mousedown` — функция, вызываемая при событии `mousedown` (нажатие по кнопке);
- `mouseup` — функция, вызываемая при событии `mouseup` (отпускание кнопки).

> ***Примечание***
>
> События `mousedown` и `mouseup` в основном используются, когда идет нажатие на кнопку, перемещение ее, а потом мышь отпускается.

Чтобы кнопка появилась на странице, ее необходимо добавить — например, так:

```
button = webiopi().createButton("bt1", "button1", fun_down, fun_up);
$("#div1").append(button);
```

Функция *WebIOPi.createFunctionButton*

Функция `WebIOPi.createFunctionButton()` создает объект — кнопку, при нажатии на которую изменяется назначение вывода GPIO.

Синтаксис: `WebIOPi.createFunctionButton(gpio)`

Параметр: `gpio` — вывод, назначение которого изменяется (`IN`, `OUT`), при этом на кнопке выводится соответствующая надпись (**IN**, **OUT**).

Функция *WebIOPi.createGPIOButton*

Функция `WebIOPi.createGPIOButton()` создает объект — кнопку, при нажатии на которую изменяется значение (0 или 1) вывода GPIO.

Синтаксис: `WebIOPi.createGPIOButton(label, gpio)`

Параметры:

- ❒ `label` — надпись на кнопке;
- ❒ `gpio` — вывод, назначение которого изменяется (`0`, `1`), при этом на кнопке выводится соответствующая надпись (**0**, **1**).

Функция *WebIOPi.createMacroButton*

Функция `WebIOPi.createMacroButton()` создает объект — кнопку, при нажатии на которую выполняется макрофункция на сервере (макрофункции прописываются в Python-скрипте, запускаемом при старте `webiopi`).

Синтаксис: `WebIOPi.createMacroButton(id, label, macro, args)`

Параметры:

- ❒ `id` — идентификатор кнопки;
- ❒ `label` — надпись на кнопке;
- ❒ `macro` — название макрофункции на сервере;
- ❒ `args` — список параметров, передаваемых макрофункции.

Функция *WebIOPi.createSequenceButton*

Функция `WebIOPi.createSequenceButton()` создает объект — кнопку, при нажатии на которую на вывод GPIO отправляется последовательность битов.

Синтаксис: `WebIOPi.createSequenceButton(id, label, gpio, period, sequence)`

Параметры:

- ❒ `id` — идентификатор кнопки;
- ❒ `label` — надпись на кнопке;
- ❒ `gpio` — вывод для отправки последовательности битов;
- ❒ `period` — время отправки одного бита (мсек);
- ❒ `sequence` — последовательность битов в виде строки.
- ❒ Пример:

```
button = webiopi().createSequenceButton("but1", "label1", 25, 100,
"01010100110011001100101010");

$("#div1").append(button);
```

Функция *WebIOPi.createRatioSlider*

Функция `WebIOPi.createRatioSlider()` создает объект — шкалу (рис. 6.15), положением указателя на которой регулируется значение ШИМ-сигнала на выводе GPIO.

Синтаксис: `WebIOPi.createRatioSlider(gpio, ratio)`

Параметры:

- ❒ `gpio` — вывод для отправки последовательности битов;
- ❒ `ratio` — начальное значение для ШИМ (0,0–1,0).

Пример:

```
button = webiopi().createRatioSlider(24, 1.0);
$("#div1").append(button);
```

Функция *WebIOPi.createAngleSlider*

Функция `WebIOPi.createAngleSlider()` создает объект — шкалу, положением указателя которой регулируется значение ШИМ-сигнала, подаваемого на вывод GPIO при управлении сервоприводом для значений поворота угла рабочего органа от –45 до +45 градусов.

Синтаксис: `WebIOPi.createAngleSlider(gpio, angle)`

Параметры:

- `gpio` — вывод для отправки последовательности битов;
- `angle` — начальное значение угла для сигнала ШИМ (от –45 до +45).

Функция *WebIOPi.setLabel*

Функция `WebIOPi.setLabel()` изменяет надпись на кнопке.

Синтаксис: `WebIOPi.setLabel(id, label)`

Параметры:

- `id` — идентификатор кнопки;
- `lable` — новая надпись для кнопки.

В листинге 6.8 представлен пример кода страницы для формирования элементов управления `webiopi`, а на рис. 6.15 представлен вид этой HTML-страницы.

Листинг 6.8. Пример кода страницы для формирования элементов управления `webiopi`

```
<html>
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta name="viewport" content = "height = device-height, width = 420,
user-scalable = no" />
        <title>WebIOPi | Demo</title>
        <script type="text/javascript" src="/webiopi.js"></script>
        <script type="text/javascript">
        webiopi().ready(function() {
        var content, button;
        content = $("#content");

        button = webiopi().createFunctionButton(25);
        content.append(button);

        button = webiopi().createGPIOButton(7, "SWITCH");
        content.append(button);
```

```
        button = webiopi().createSequenceButton("sos", "S.O.S 1", 25, 100,
"01010100110011001100101010");
        content.append(button);

        button = webiopi().createMacroButton("macro", "Print Time",
"PrintTime");
        content.append(button);

        button = webiopi().createAngleSlider(23, 30);
        content.append(button);

        button = webiopi().createRatioSlider(24, 0.5);
        content.append(button);

        webiopi().refreshGPIO(true);
        });

</head>
<body>
        <div id="content" align="center"></div>
</body>
</html>
```

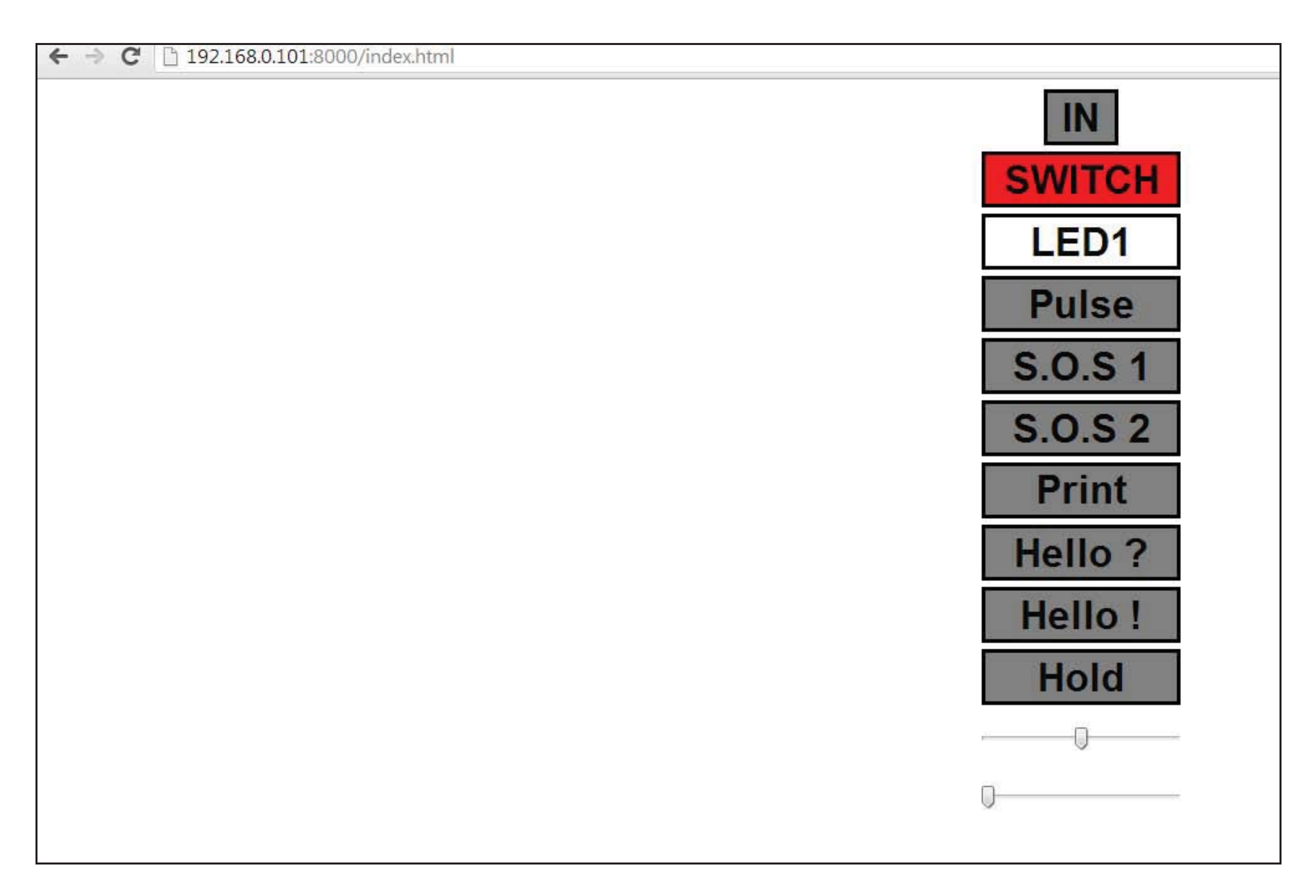


Рис. 6.15. Вид HTML-страницы с элементами управления `webiopi`

6.2.4. Проект управления веб-камерой на сервоприводах

Для создания своего проекта с использованием фреймворка WebIOPi необходимо реализовать следующие его компоненты:

- веб-интерфейс — HTML-страница с использованием библиотеки webiopi.js;
- серверный скрипт на языке Python, запускаемый при старте `webiopi` для реализации функционала веб-интерфейса;
- файл конфигурации с установкой начальных значений портов.

Подготовим проект управления из веб-интерфейса камерой, укрепленной на подвесе, снабженном сервоприводами. Из веб-интерфейса можно будет поворачивать камеру, а также делать снимки и отправлять их на электронную почту.

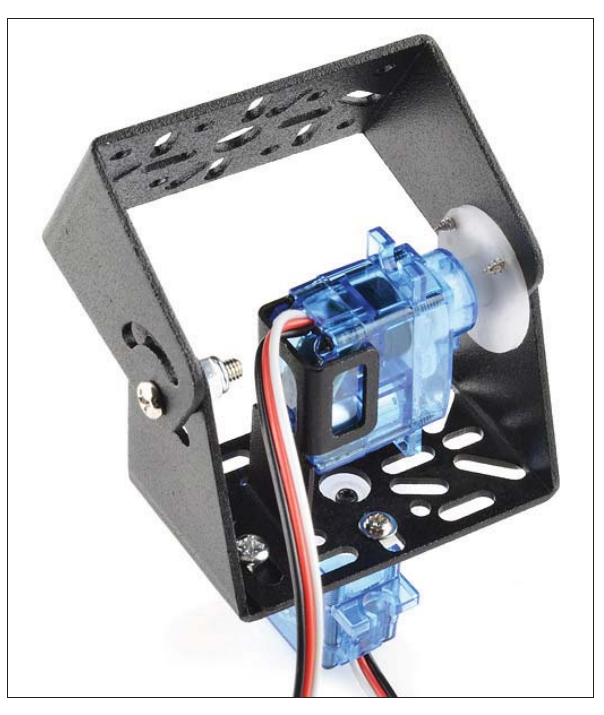


Рис. 6.16. Подвес для камеры с сервоприводами



Рис. 6.17. Сервопривод TG9

Для проекта понадобятся следующие детали:

- подвес для камеры (рис. 6.16);
- сервоприводы TG9 (рис. 6.17) — 2 шт.;
- дополнительный блок питания 5 В;
- камера Raspberry Camera Board (см. рис. 3.35).

Для управления сервоприводами задействуем два вывода GPIO (GPIO23 и GPIO24). Электрическая схема проекта представлена на рис. 6.18.

Сначала настроим в файле конфигурации config, который находится в папке /etc/webiopi/, необходимые параметры: путь к домашней папке и путь к скрипту, выполняемому при запуске `webiopi`:

```
doc-root = /home/pi/webiopi/examples/servo-camera
script = /home/pi/webiopi/examples/servo-camera/camera.py
```

В *разд. 4.11.2* мы уже рассматривали передачу потокового видео с камеры Raspberry Pi Camera Board. Немного изменим скрипт запуска камеры (файл stream_start.sh) — картинку, получаемую с камеры, лучше сохранять в папке нашей домашней страницы. Содержимое измененного файла stream_start.sh представлено в листинге 6.9.

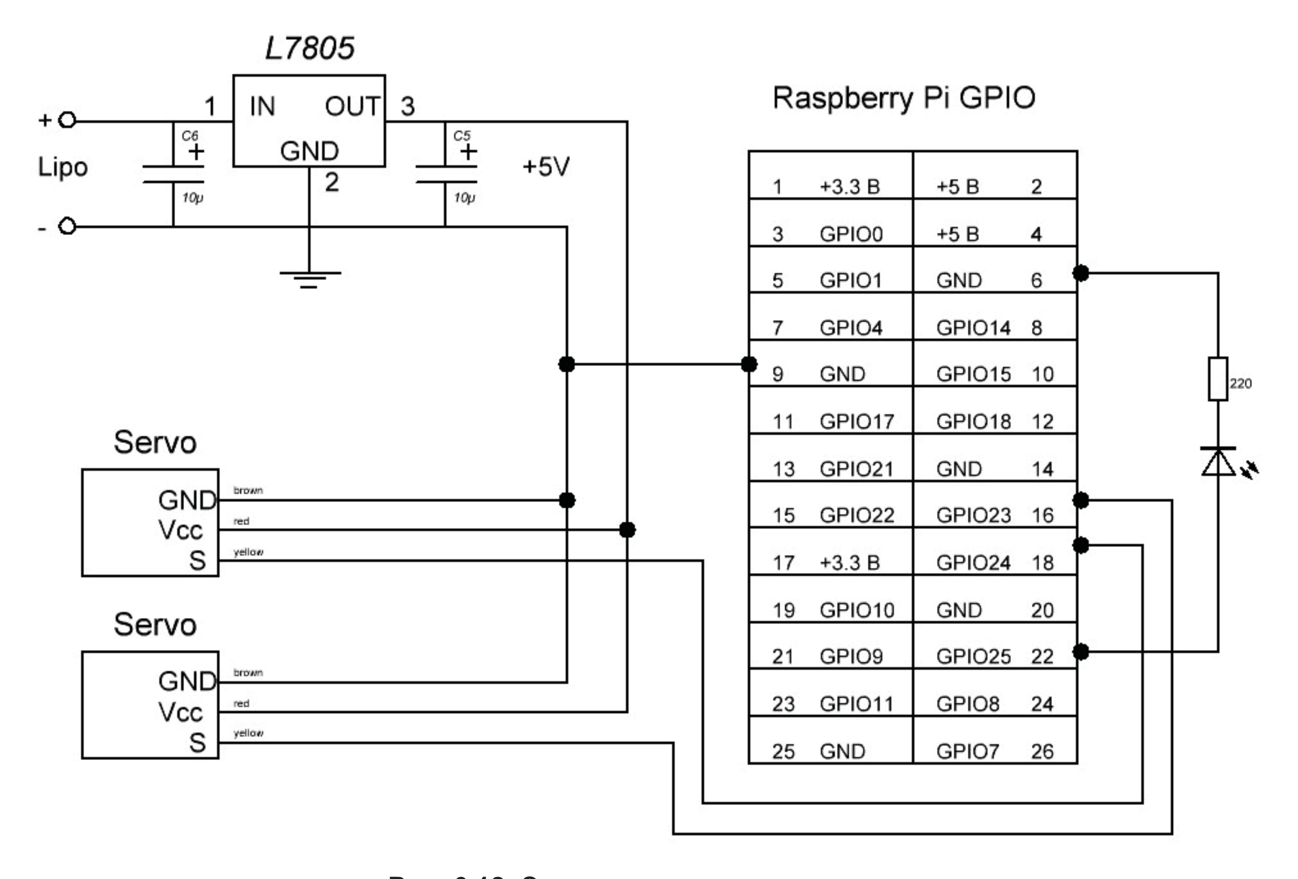


Рис. 6.18. Электрическая схема проекта

Листинг 6.9. Файл stream_start.sh

```
#!/bin/bash
if [ -d /tmp/stream ];then
        echo "/tmp/stream already created"
else
        mkdir /tmp/stream
fi

if [ -f /tmp/stream/pic.jpg ];then
        echo "raspistill already running"
else
        raspistill -w 320 -h 240 -q 5 -o /home/pi/webiopi/examples/servo-
camera/pic.jpg -tl 100 -t 9999999&
fi
mjpg_streamer -i "input_file.so -f /home/pi/webiopi/examples/servo-camera"
-o "output_http.so -w /home/pi/webiopi/examples/servo-camera"
```

Напишем на Python серверный скрипт camera.py, который будет запускаться при запуске `webiopi`. В скрипте пропишем назначение контактов, макросы и выполним начальный запуск камеры потокового видео. Содержимое скрипта camera.py представлено в листинге 6.10.

Управляющие контакты сервоприводов подключены к выводам 16 (GPIO23) и 18 (GPIO24) GPIO. Плюс еще одна дополнительная опция — мигание светодиода, подключенного к выводу 22 (GPIO25).

Листинг 6.10. Серверный скрипт camera.py

```
import webiopi
import sys
import time
from subprocess import call

GPIO = webiopi.GPIO

SERVO1  = 23
SERVO2  = 23
LED1   = 25

def camera_start():
    return_code = call("/usr/share/webiopi/examples/servo-
camera/stream_start.sh", shell=True)

def setup():
    webiopi.debug("Blink script - Setup")
    # Установка выводов GPIO
    GPIO.setFunction(SERVO1, GPIO.PWM)
    GPIO.setFunction(SERVO1, GPIO.PWM)
    GPIO.setFunction(LED1, GPIO.OUT)
    GPIO.pwmWriteAngle(SERVO1, 0)    # set to 0 (neutral)
    GPIO.pwmWriteAngle(SERVO2, 0)    # set to 0 (neutral)
    GPIO.digitalWrite(LED1, GPIO.HIGH)
    camera_start()

# Цикл loop()
def loop():
    # Toggle LED each 5 seconds
    value = not GPIO.digitalRead(LED1)
    GPIO.digitalWrite(LED1, value)
    webiopi.sleep(5)

# Возврат выводов в начальное состояние
def destroy():
    webiopi.debug("Blink script - Destroy")
```

```
    # Reset GPIO functions
    GPIO.setFunction(SWITCH, GPIO.IN)
    GPIO.setFunction(SERVO, GPIO.IN)
    GPIO.setFunction(LED0, GPIO.IN)
    GPIO.setFunction(LED1, GPIO.IN)
```

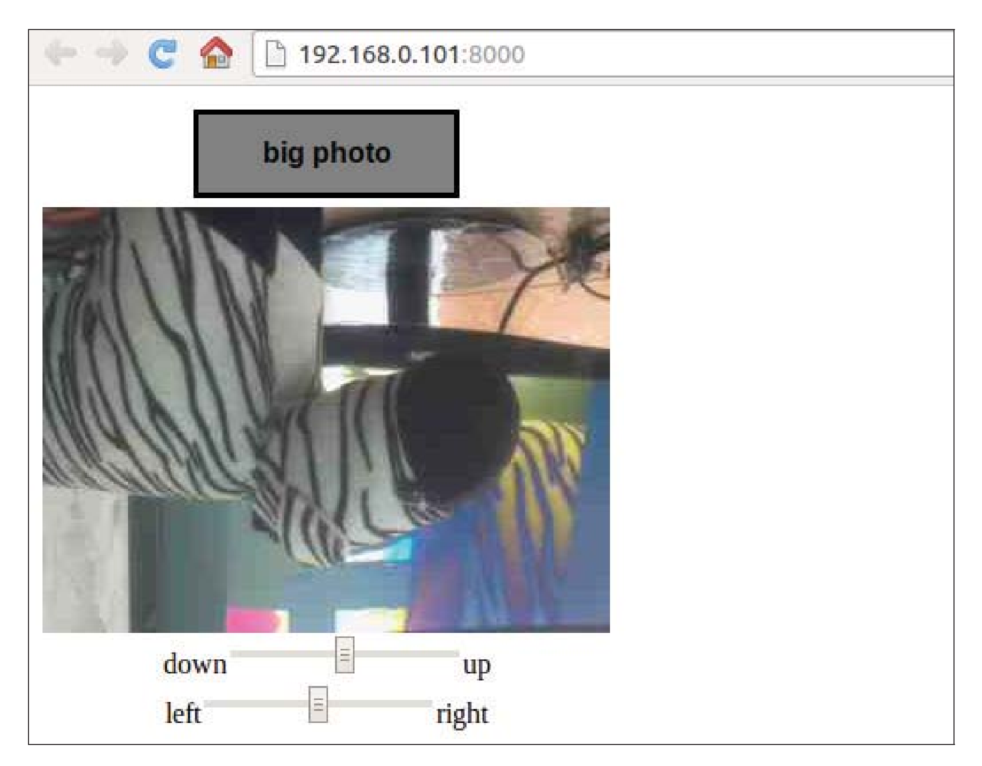


Рис. 6.19. HTML-страница проекта управления подвесом для камеры

Веб-интерфейс у нас будет достаточно простой: два слайдера управления подвесом камеры, кнопка для отправки фото на email и картинка изображения с камеры (рис. 6.19). При изменении позиции слайдера подвес поворачивается в направлении вверх-вниз и влево-вправо. Изображение с камеры постоянно обновляется запуском функции `set_Timeout()`. Необходимо также запретить кэширование страницы. Код HTML-страницы camera.html представлен в листинге 6.11.

Листинг 6.11. Код HTML-страницы camera.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
       <meta http-equiv = "cache-control" content = "max-age=0">
       <meta http-equiv = "cache-control" content = "no-cache">
       <meta http-equiv = "expires" content = "0">
       <meta http-equiv = "expires" content = "Tue, 01 Jan 1970 1:00:00 GMT">
       <meta http-equiv = "pragma" content = "no-cache">
       <meta name="viewport" content = "height = device-height, width = 420,
user-scalable = no" />
```

```
        <title>WebIOPi camera</title>
        <script type="text/javascript" src="/webiopi.js"></script>
        <script type="text/javascript">
        function init() {
                var button;

                  button = webiopi().createButton("photo", "big photo", photo);
                  $("#cam").append(button);

                  button = webiopi().createAngleSlider(23, 30);
                  $("#updown").append(button);


                  button = webiopi().createAngleSlider(24, 0);
                  $("#leftright").append(button);
                  }

        function photo() {
                 $("#ph").html('');
                 $("#ph").html('<img src="pic.jpg">');
                 }
        function get_img()
          {
document.getElementById("img1").src="http://192.168.0.101:8080/pic.jpg";
          }
        webiopi().ready(init);

        </script>
        <style type="text/css">
                button {
                        margin: 5px 5px 5px 5px;
                        width: 150px;
                        height: 50px;
                        font-size: 12pt;
                        font-weight: bold;
                        color: black;
                }
        </style>
</head>
<body onload='setInterval("get_img();",1000);'>
        <div style="float:left">
        <div id="content" align="center">
                 <div id="cam"></div>
                <div id="vid"><img id="img1" width="320" height="240"
src="http://192.168.0.101:8080/pic.jpg"></div>
                <div>down<span id="updown"></span>up</div>
```

```
            <div>left<span id="leftright"></span>right</div>
        </div>
        </div>
        <div style="float:right" id="ph"></div>
</body>
</html>
```

> ***Примечание***
>
> Коды листингов этого проекта вы найдете в папке *glava_06\servo-camera* сопровождающего книгу электронного архива (см. *приложение*).

6.2.5. WebIOPi — подключение устройств

Фреймворк WebIOPi позволяет использовать устройства типов Serial, I^2C, SPI и 1-Wire прямо из REST API, без написания макросов. Список поддерживаемых датчиков имеется на странице: **https://code.google.com/p/webiopi/wiki/DEVICES**.

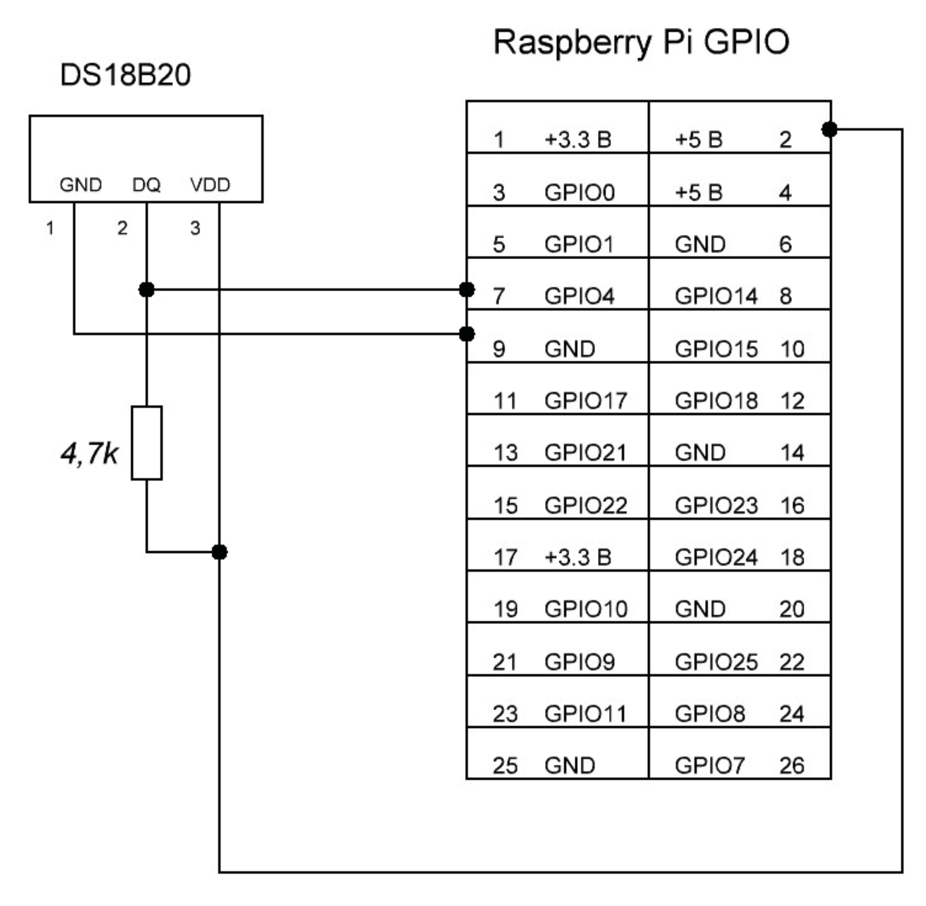


Рис. 6.20. Схема подключения датчика DS18B20

Включить поддержку устройства можно в блоке `[DEVICE]` файла конфигурации /etc/webiopi/config, раскомментировав строку с выбранным датчиком:

```
[DEVICES]
...
#temp0 = TMP102
#temp1 = TMP102 slave:0x49
```

```
temp2 = DS18B20
#temp3 = DS18B20 slave:28-0000049bc218
```

Датчик следует подключить к выходам GPIO. Пример схемы подключения для датчика DS18B20 представлен на рис. 6.20.

Выполнив необходимые монтажные манипуляции, открываем стартовую страницу WebIOPi (см. рис. 6.11), выбираем пункт **Device Monitor** и попадаем на страницу подключенных устройств (рис. 6.21). В нашем случае здесь отображен только датчик температуры DS18B20. Страница демонстрирует данные, получаемые с датчика, с обновлением их каждые 5 секунд.

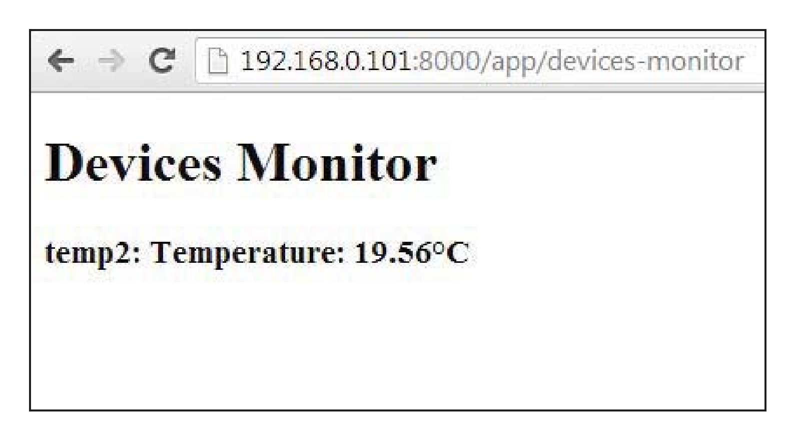


Рис. 6.21. Монитор подключенных устройств

6.3. Плата расширения Gertboard

Устройство Gertboard представляет собой плату расширения портов ввода/вывода (I/O) для микрокомпьютера Raspberry Pi. Плата подключается к порту GPIO (General Purpose Input/Output) с помощью двойного ряда контактов, расположенного в верхней левой части Raspberry Pi, и разъема на задней панели платы Gertboard (рис. 6.22).

При соединении этих разъемов необходимо соблюдать осторожность, т. к. очень легко промахнуться и подключить только один ряд штырьков, что может привести к отказу портов GPIO как на плате Gertboard, так и на борту Raspberry Pi. Плата Gertboard получает питание через контакты порта GPIO, так что для Raspberry Pi потребуется источник питания, рассчитанный на ток не менее 1,5 А.

Плата Gertboard имеет набор функциональных блоков (основные возможности платы), которые могут быть соединены множеством способов с различными внешними устройствами с помощью штырьков-контактов.

Функциональные блоки:

- 12 буферизированных портов ввода/вывода с индикацией;
- 3 кнопки управления;
- 6-канальный драйвер с открытым коллектором (50 V, 0,5 А);
- контроллер двигателя постоянного тока (18 V, 2 А);
- 28-контактный двунаправленный порт микроконтроллера ATmega;

- 2-канальный 8-, 10- или 12-битный цифроаналоговый преобразователь (D\A);
- 2-канальный 10-битный аналого-цифровой преобразователь (A\D).

Расположение блоков на плате Gertboard ревизии 2 показано на рис. 6.23.



Рис. 6.22. Плата Gertboard, подключенная к Raspberry Pi



Рис. 6.23. Функциональные блоки платы Gertboard

Между функциональными блоками платы Gertboard нет электрических соединений кроме цепей питания и заземления. Поэтому для создания нужных соединений между различными функциональными блоками на плате используются дополнительные проводники с разъемами и перемычки (рис. 6.24), входящие в комплект поставки.

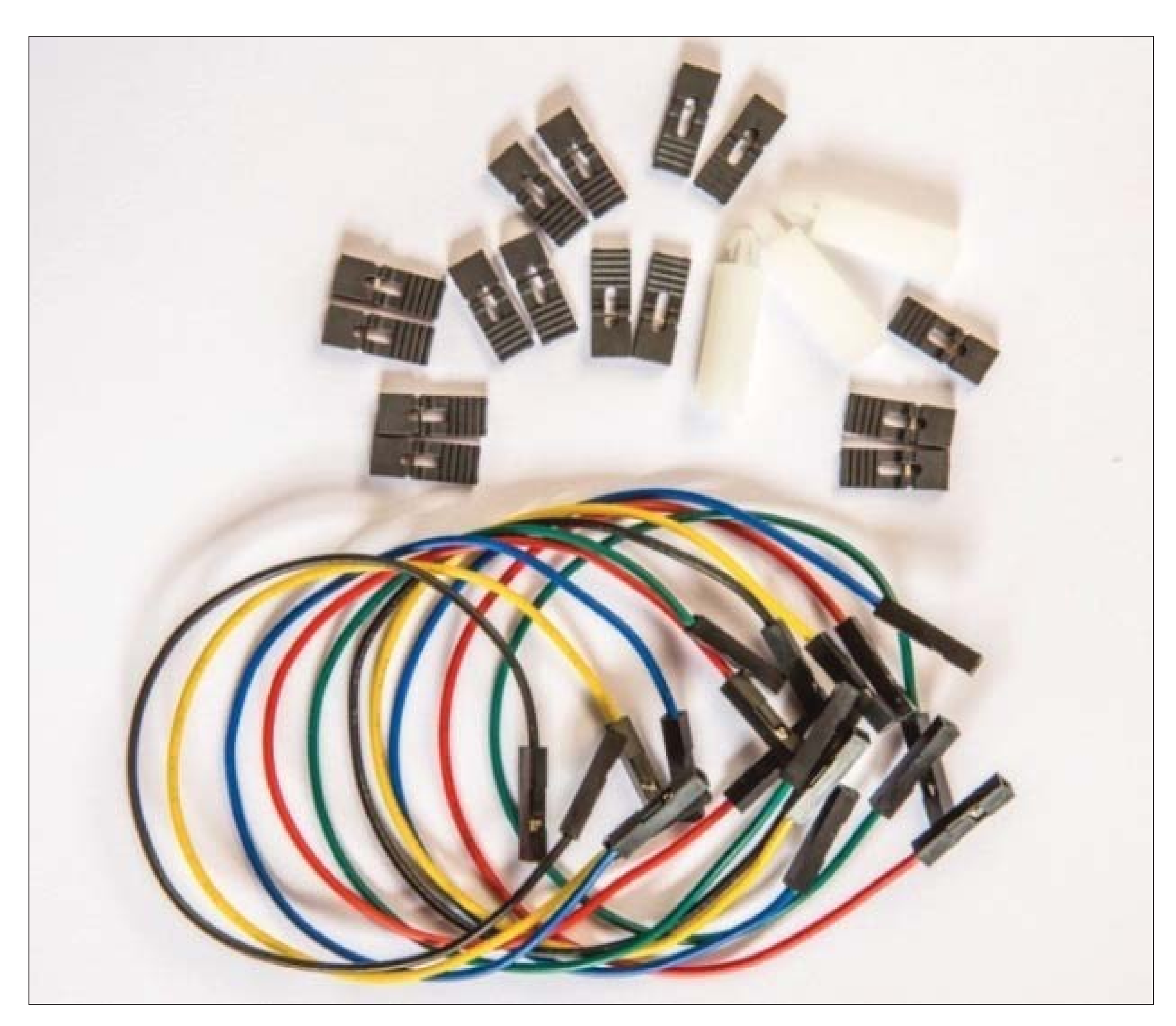


Рис. 6.24. Дополнительные проводники и перемычки для платы Gertboard

6.3.1. Включение платы Gertboard

Выводы питания обозначены на плате соответственно значениям их напряжений — например, 5V или 3V3. Питание напряжением 5V приходит на плату через разъем GPIO от Raspberry Pi, и, если нужно получить это напряжение дополнительно, его можно снять со штырька J24, располагающегося ближе к нижнему правому углу платы. Плюс отмечен меткой 5V. Общий провод или минус отмечен на плате меткой GND. Напряжение питания +3,3V — это напряжение, получаемое из тех же 5V, которые приходят на плату через разъем J1. Для подачи этого напряжения компонентам платы Gertboard необходимо установить перемычку на два верхние штырька переключателя J7. Он находится ближе к нижнему правому углу платы (см. фото

и схему на рис. 6.25). Драйвер с открытым коллектором и контроллер двигателя могут работать на более высоких напряжениях, что потребует подключения внешнего источника питания соответствующего напряжения.

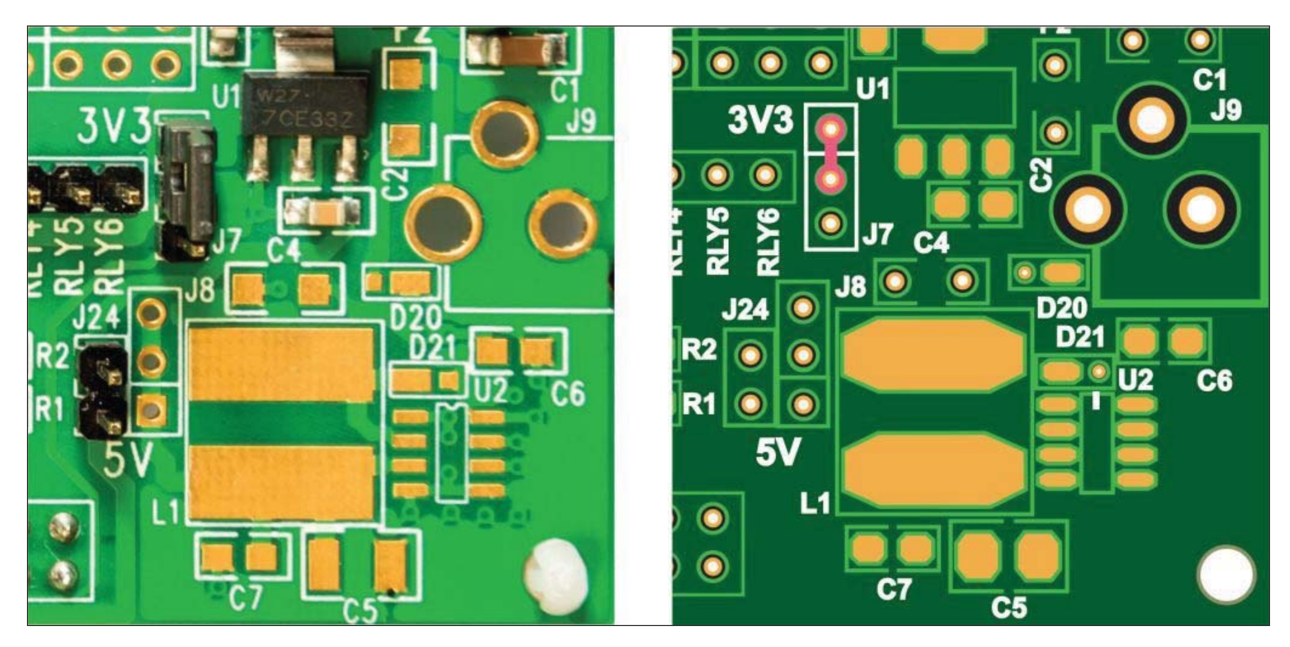


Рис. 6.25. Перемычка для переключения на питание 3,3 В

6.3.2. Контакты портов GPIO

Штырьки разъема J2 (справа от надписи **Raspberry Pi Gertboard** на плате) обеспечивают доступ ко всем портам ввода/вывода в разъеме GPIO. На нижней панели платы имеется двухрядный 26-контактный разъем J1 (см. рис. 6.1), который непосредственно или через 26-проводной шлейф соединяет плату Gertboard и компьютер Raspberry Pi. Однако только 17 контактов в разъеме J2 задействованы на ввод/вывод: три контакта в разъеме J1 являются питающими 3,3V, 5V и GND, а шесть не задействованы — DNC (Do Not Connect, не подключены). И поэтому метки на этих выводах GP0, GP1, GP4, GP7 и т. д. могут поначалу показаться произвольными, поскольку нумерация их произведена не по порядку, и цифры не совпадают с номерами штырьков разъема J1 GPIO. Но тут важно, что эти метки соответствуют сигналам и именам, которые использует процессор BCM2835, установленный на борту Raspberry Pi. Сигнал порта GPIO*n* в техническом описании (Datasheet) процессора BCM2835 соответствует контакту GP*n* в разъеме J2. По крайней мере, это было верно для первой ревизии Raspberry Pi ("rev1"). С сентября 2012 года начались продажи второй ревизии Raspberry Pi ("Rev2"). И на этих устройствах некоторые из выводов порта GPIO были изменены. Порт I/O с номером GPIO21 в настоящее время несет номер GPIO27, а порты, которые раньше были с номерами GPIO0 и GPIO1, в настоящее время пронумерованы как GPIO2 и GPIO3. Эти изменения представлены в табл. 6.1. Некоторые из выводов GPIO осуществляют альтернативные функции, что также отражено в табл. 6.1.

Таблица 6.1. *Соответствие выводов плат Gertboard и Raspberry Pi (RPi)*

Gertboard	**RPi rev. 1**	**RPi rev. 2**	**Альтернативная функция**	**Цепь**
GP 0	GPIO0	GPIO2	SDA	I^2C
GP 1	GPIO1	GPIO3	SCL	I^2C
GP 4	GPIO4	GPIO4		
GP 7	GPIO7	GPIO7	SPI_CE1_N	SPI
GP 8	GPIO8	GPIO8	SPI_CE0_N	SPI
GP 9	GPIO9	GPIO9	SPI_MISO	SPI
GP 10	GPIO10	GPIO10	SPI_MOSI	SPI
GP 11	GPIO11	GPIO11	SPI_SCLK	SPI
GP 14	GPIO14	GPIO14	TXD 0	UART
GP 15	GPIO15	GPIO15	RXD 0	UART
GP 17	GPIO17	GPIO17		
GP 18	GPIO18	GPIO18	PWM 0	ШИМ
GP 21	GPIO21	GPIO27		
GP 22	GPIO22	GPIO22		
GP 23	GPIO23	GPIO23		
GP 24	GPIO24	GPIO24		
GP 25	GPIO25	GPIO25		

6.3.3. Тестовые программы для Gertboard

Для платы Gertboard существует множество тестовых программ, написанных на языках программирования C и Python. Язык C обеспечивает наиболее полный доступ к функциональным возможностям платы Gertboard, но этот язык труден для восприятия начинающими программистами, поэтому многие программы написаны на Python — такие, например, как пакеты для доступа к портам GPIO на Raspberry Pi, для включения альтернативных функций этих контактов, для доступа к последовательному периферийному интерфейсу (SPI) и широтно-импульсному модулятору (ШИМ, PWM) и др. Так что, используя пакеты, написанные на языке Python, можно получить доступ к большинству функциональных возможностей платы Gertboard. Однако микроконтроллер Atmel ATmega запрограммировать без знания языка C вы не сможете.

Далее мы рассмотрим программное обеспечение на языке Python 2.7, созданное, наряду с примерами и документацией по этому языку, Алексом Эймсом (Alex Eames), ведущим блога RasPi.TV на канале YouTube. Это программное обеспече-

ние совместимо со всеми текущими ревизиями платы Gertboard и компьютера Raspberry Pi.

Для загрузки программного обеспечения на компьютер Raspberry Pi сначала создадим для него соответствующий каталог, перейдем в него и выполним команду:

```
sudo mkdir gertboard_python
sudo chmod 0777 gertboard_python
cd gertboard_python
wget http://raspi.tv/download/GB_Python.zip
```

Затем распакуем в этот каталог полученный архив:

```
unzip GB_Python.zip
cd GB_Python
ls
```

Команда `ls` выведет на экран список всех размещенных в каталоге файлов. Большинство из них — файлы программ на языке Python. Для управления GPIO на Python существуют два пакета общего назначения: RPi.GPIO и WiringPi. Программы, которые поставляются в двух вариантах, — например, leds-rg.py и leds-wp.py, запускаются каждая с помощью своего пакета. Желательно, чтобы в комплекте присутствовали оба пакета, потому что ни один из них все еще не имеет полностью законченного набора возможностей, но в большинстве случаев возможности одного перекрывают недостатки другого. Так, слабостью пакета RPi.GPIO является отсутствие поддержки аппаратного PWM (широтно-импульсного модулятора), используемого в программе для управления двигателями. Слабостью пакета WiringPi является отсутствие возможности управления подтягивающими резисторами, необходимыми для правильного использования кнопок. Программы, ориентированные на пакет RPi.GPIO, вызываются командой: `*filename*-rg.py`, а те, которые используют пакет WiringPi, — командой: `*filename*-wp.py`.

Вот список всех тестовых программ:

- buttons-rg.py — программа кнопок для RPi.GPIO;
- leds-rg.py — программа светодиодов для RPi.GPIO;
- leds-wp.py — программа светодиодов для WiringPi;
- butled-rg.py — программа кнопок и светодиодов для RPi.GPIO;
- motor-rg.py — программа мотора с использованием программного PWM и RPi.GPIO;
- motor-wp.py — программа мотора с использованием аппаратного PWM и WiringPi;
- ocol-rg.py — программа переключения реле для RPi.GPIO;
- ocol-wp.py — программа переключения реле для WiringPi;
- atod.py — тест аналого-цифрового преобразователя (A/D) с использованием SPI и модуля `spidev`;
- dtoa.py — тест цифроаналогового преобразователя (D/A) с использованием SPI и модуля `spidev`;

- dad.py — тест для обоих преобразователей (D/A и A/D) с использованием SPI и модуля `spidev`;
- potmot.py — тест с A/D и двигателем с использованием WiringPi и модуля `spidev`.

Буферизированные порты ввода/вывода

В качестве входных или выходных портов могут использоваться 12 штырьков-контактов, и каждый из них может быть назначен входным либо выходным с помощью перемычки.

Рассмотрим как термины "вход" и "выход" применяются к платам Gertboard и Raspberry Pi. В режиме входа с Raspberry Pi на плату Gertboard через порты GPIO осуществляется вывод данных, а в режиме выхода с платы Gertboard Raspberry Pi должен быть настроен на ввод данных. Важно помнить об этом и не путать установку перемычек платы Gertboard — при включенном режиме выхода из Gertboard порты на Raspberry Pi включаются на вход.

На рис. 6.26 показана схема установки перемычек портов I/O платы Gertboard с 4 по 12.

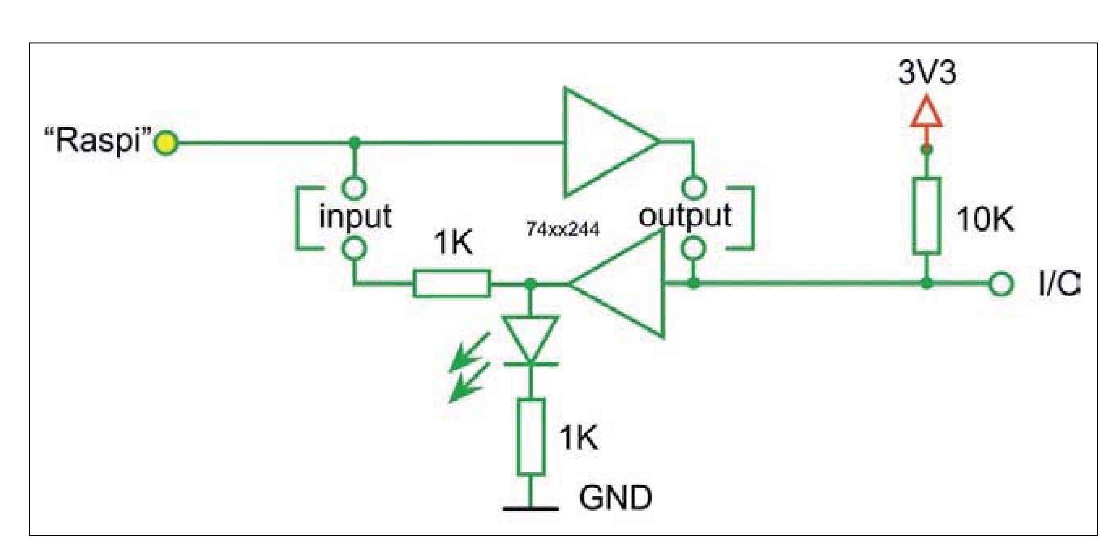


Рис. 6.26. Схема установки перемычек портов I/O с 4 по 12

Треугольными символами на схеме обозначены буферы, которые выдают логические значения низкого и высокого уровней в направлении, указываемом треугольником. Чтобы порты функционировали в качестве входа, установите входную перемычку в Raspberry Pi, как показано на схеме с пометкой **input**. При этом поток данных идет от платы к компьютеру. Чтобы порты функционировали в качестве выхода, необходимо установить выходную перемычку **output**, и поток данных пойдет от Raspberry Pi к плате. Если по ошибке или иной причине будут установлены сразу обе перемычки, то это не вызовет неисправности, но порт не будет правильно работать.

Светодиоды

Во входных и выходных режимах светодиод показывает, что на штырьке-выводе I/O присутствует логический уровень. Светодиод будет светиться, если высокий

уровень присутствует на выводе, и не будет светиться при низком уровне. Существует и третий вариант для использования этого порта — когда ни одна из входных и выходных перемычек не установлена. В этом случае порт может быть использован в качестве простого логического детектора, который подключается к любой активной точке с напряжением 0 В либо 3,3 В, а индикатор покажет, какой уровень будет в выбранной точке.

Резистор 10 кОм, включенный в цепь питания 3V3 (см. на рис. 6.26 справа) — подтягивающий. Если бы его не было, светодиод включался бы при любой минимальной электрической наводке или при случайном касании рукой вывода. Установка подтягивающего резистора предотвращает такое нештатное поведение.

Между входным буфером и выводами GPIO Raspberry Pi включен ряд ограничивающих резисторов, что необходимо для защиты процессора BCM2835, установленного на Raspberry Pi, в случае, если пользователь случайно оставит входную перемычку на месте, а с борта компьютера придет управляющий сигнал, посланный запущенной программой. Тогда нагрузкой вывода процессора BCM2835 и окажется ограничивающий резистор 1 кОм, и никакие элементы не выйдут из строя.

Кнопки

На плате Gertboard расположены три кнопки, присоединенные к портам 1, 2 и 3. Схема этих портов показана на рис. 6.27. Эта схема, по существу, та же, что и на рис. 6.26, только с добавлением кнопочного переключателя и резистора с левой стороны. Когда кнопка нажата, вывод порта Raspberry Pi подключается к "земле" через резистор и получает низкий логический уровень.

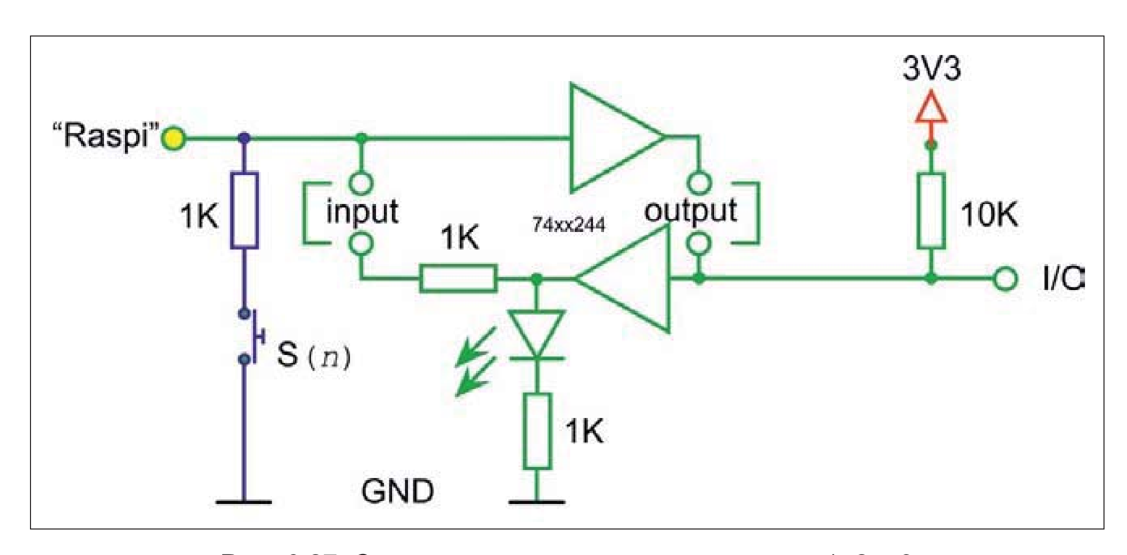


Рис. 6.27. Схема подключения кнопок к портам 1, 2 и 3

Для того чтобы использовать кнопки на вход, в Raspberry Pi входная перемычка не устанавливается. Если установлена перемычка на выходе из нижнего по схеме буфера, то кнопка правильно работать не будет. Чтобы наглядно видеть работу кнопки, можно установить выходную перемычку, и тогда светодиод будет загораться при нажатии. Для правильной работы кнопок на борту Raspberry Pi должны быть включены подтягивающие резисторы. Если кнопки не нажаты, то с порта GPIO считывается высокий логический уровень, а при нажатии — низкий.

Тестирование кнопок

Для проверки работы кнопок существуют тестовые программы — например, программа на Python под названием buttons-rg.py. Чтобы запустить этот тест на плате Gertboard, должны быть установлены соединения, как показано на рис. 6.28.

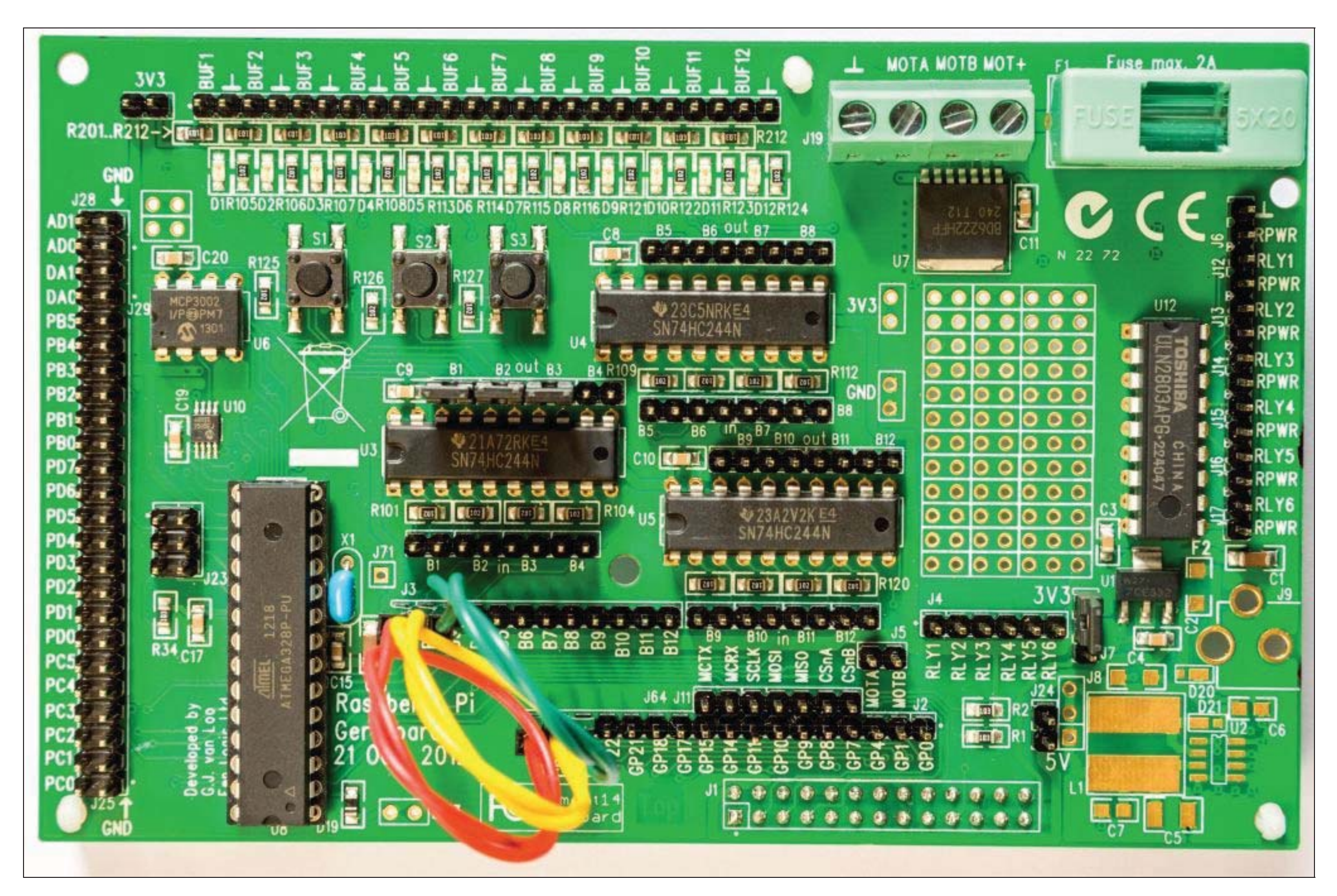


Рис. 6.28. Соединения проводников для теста кнопок

Так, необходимо соединить проводниками и перемычками штырьки-контакты B1, B2, B3 в разъеме J3 и контакты портов GPIO: GP25, GP24 и GP23 — в разъеме J2 соответственно. Таким образом GPIO25 будет считывать состояние левой кнопки, GPIO24 — состояние средней, а GPIO23 — состояние правой. Перемычки на стороне "out" буфера U3: U3-out-B1, U3-out-B2 и U3-out-B3 — не являются обязательными, но если они будут установлены, левые три светодиода загорятся при нажатии на кнопки и укажут текущее состояние кнопок.

Программа buttons-rg.py доступна только при использовании пакета RPi.GPIO (на данный момент функция включения подтягивающих резисторов в WiringPi для Python пока недоступна, а без этого кнопки не будут работать правильно). Содержимое файла buttons-rg.py представлено в листинге 6.12.

Листинг 6.12. Программа buttons-rg.py

```
import RPi.GPIO as GPIO
import sys
board_type = sys.argv[-1]
```

```
GPIO.setmode(GPIO.BCM)                  # инициализация RPi.GPIO
for i in range(23,26):                  # использовать порты 23-25
    GPIO.setup(i, GPIO.IN, pull_up_down=GPIO.PUD_UP)  # установить
подтягивающие резисторы

# сообщения в терминал
if board_type == "m":
    print "These are the connections for the Multiface buttons test:"
    print "GPIO 25 --- 1 in BUFFERS"
    print "GPIO 24 --- 2 in BUFFERS"
    print "GPIO 23 --- 3 in BUFFERS"
    print "Optionally, if you want the LEDs to reflect button state do the
following:"
    print "jumper on BUFFER DIRECTION SETTINGS, OUT 1"
    print "jumper on BUFFER DIRECTION SETTINGS, OUT 2"
    print "jumper on BUFFER DIRECTION SETTINGS, OUT 3"

else:
    print "These are the connections for the Gertboard buttons test:"
    print "GP25 in J2 --- B1 in J3"
    print "GP24 in J2 --- B2 in J3"
    print "GP23 in J2 --- B3 in J3"
    print "Optionally, if you want the LEDs to reflect button state do the
following:"
    print "jumper on U3-out-B1"
    print "jumper on U3-out-B2"
    print "jumper on U3-out-B3"

raw_input("When ready hit enter.\n")

button_press = 0
previous_status = ''

try:
    while button_press < 20:
        status_list = [GPIO.input(25), GPIO.input(24), GPIO.input(23)]
        for i in range(0,3):
            if status_list[i]:
                status_list[i] = "1"
            else:
                status_list[i] = "0"
        # dump current status values in a variable
        current_status =
''.join((status_list[0],status_list[1],status_list[2]))
        if current_status != previous_status:
            print current_status
            previous_status = current_status
            button_press += 1                   counter
```

```
except KeyboardInterrupt:           # CTRL+C keyboard interrupt
    GPIO.cleanup()
GPIO.cleanup()                      # при выходе сбросить порты
```

Сначала программа импортирует модуль `RPi.GPIO`, необходимый для обработки портов GPIO. Затем команда:

```
GPIO.setmode (GPIO.BCM)
```

устанавливает схему нумерации для контактов, соответственно нумерации выводов процессора BCM. В результате номер штырька-контакта в коде Python становится числом, которое процессор BCM2835 на борту Raspberry Pi использует для обозначения контактов. В противном случае номера в коде Python будут ссылаться на штырек-контакт с номером в разъеме J1 — таким же, как номер его размещения в разъеме P1 на Raspberry Pi.

Следующие две строки устанавливают в портах 23–25 подтягивающие резисторы:

```
for i in range(23,26):
  GPIO.setup(i, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

Как только пользователь подтверждает выданную ему информацию, устанавливаются начальные значения для переменных `button_press` и `previous_status`. Следующий цикл — `while` — выполняется до тех пор, пока не произойдет 19 нажатий на кнопку. Для каждого круга этого цикла состояние каждой кнопки считывается и сохраняется в списке `status_list`. Если кнопка не была нажата, то в файле сохраняется значение 0, если кнопка была нажата — сохраняется значение 1. Все величины файла `status_list` проверяются на соответствие в `previous_status`. Если изменить эту строку так:

```
if current_status != previous_status:
```

то выполнится небольшая часть программы, которая отобразит новые значения на экране и увеличит значение итератора `button_press`. И цикл начнется с начала.

Цикл `while` заключен в блоке `try`:

```
try:
<while block>
except KeyboardInterrupt:
```

Это даст возможность корректного выполнения команды сброса портов, если ранее была нажата комбинация клавиш <Ctrl>+<C>. При нормальном же завершении программы после 19 нажатий кнопки порты будут сброшены в любом случае. Однако если бы вы не включили в блок `try` прерывание с клавиатуры `except`, то нажатие комбинации клавиш <Ctrl>+<C> закрыло бы программу, но оставило порты открытыми. И если бы вы попытались использовать порты снова, это дало бы ошибку, т. к. в них уже имелись бы какие-то значения.

Тестирование светодиодов

Программы на языке Python тестирования светодиодов называются leds-rg.py и leds-wp.py. При настройке платы Gertboard для выполнения этого теста необходимо

установить проводники и перемычки согласно схеме, приведенной на рис. 6.29. Каждый порт I/O подключен в качестве выходного, так что все перемычки устанавливаются как OUT. Проводники используются для подключения всех контактов GP, располагающихся в разъеме J2, и всех контактов B в разъеме J3: GP25 соединяем с B1, GP24 — с B2, GP23 — с B3, GP22 — с B4, GP21 — с B5, GP18 — с B6, GP17 — с B7, GP11 — с B8, GP10 — с B9, GP9 — с B10, GP8 — с B11 и GP7 — с B12. Все 12 контактов GP слева соединены с контактами B за исключением пропущенных GP14 и GP15. Эти два контакта используются в Linux в режиме UART, так что лучше их в данной операции не задействовать.

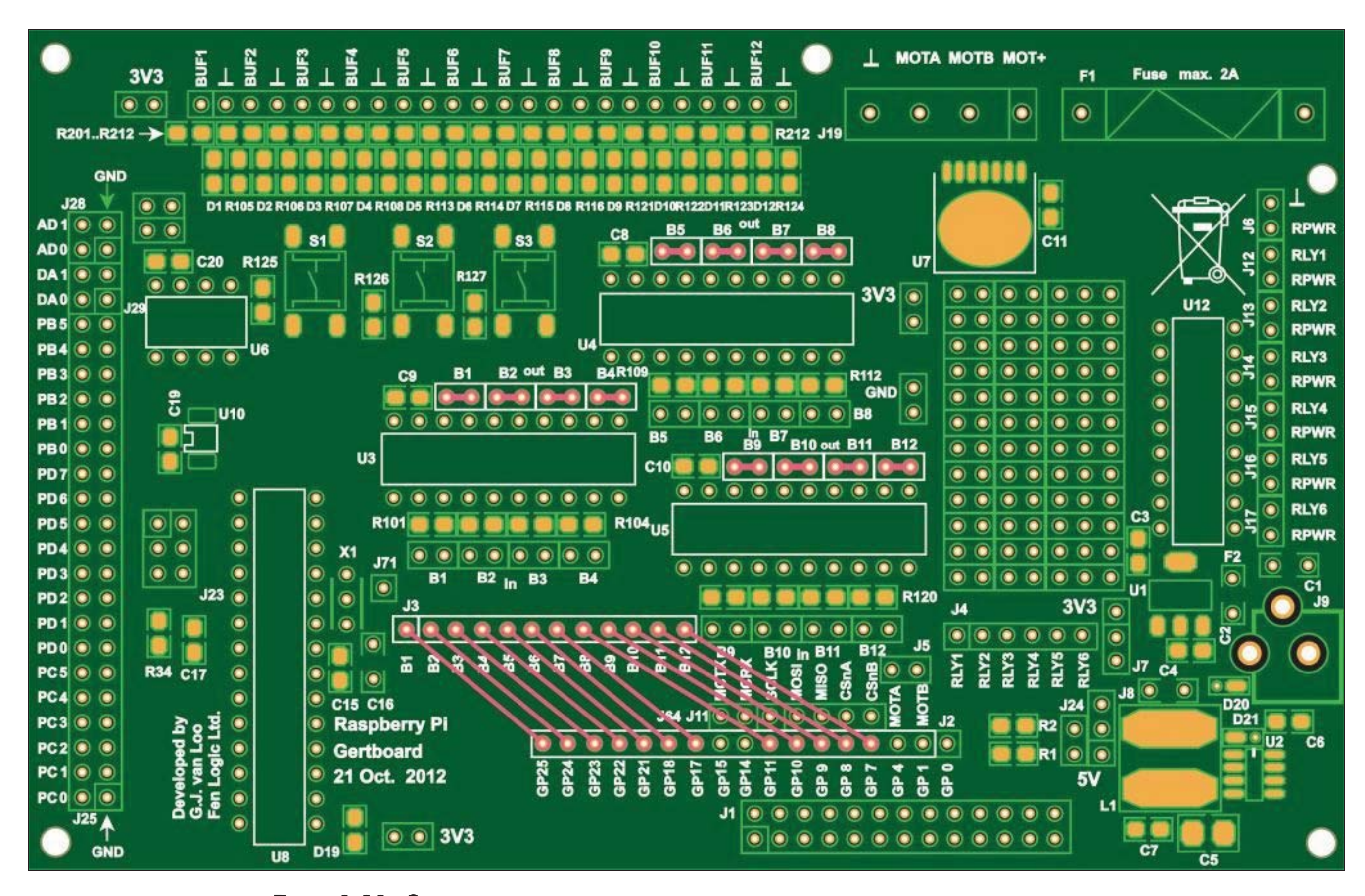


Рис. 6.29. Схема соединения проводников для теста светодиодов

Тест светодиодов доступен как для RPi.GPIO, так и для WiringPi на языке Python. Единственное различие между программами leds-rg.py и leds-wp.py — это способ нумерации выводов портов GPIO в разных ревизиях Raspberry Pi. Рассмотрим скрипт leds-rg.py, содержимое которого представлено в листинге 6.13.

Листинг 6.13. Скрипт leds-rg.py

```
import RPi.GPIO as GPIO
from time import sleep
import sys
board_type = sys.argv[-1]

if GPIO.RPI_REVISION == 1:       # получить номер ревизии платы correctly
    # установить порты для Revision 1 Pi
    ports = [25, 24, 23, 22, 21, 18, 17, 11, 10, 9, 8, 7]
```

```
else:
    ports = [25, 24, 23, 22, 27, 18, 17, 11, 10, 9, 8, 7]
ports_rev = ports[:]
ports_rev.reverse()

GPIO.setmode(GPIO.BCM)           # инициализация RPi.GPIO

for port_num in ports:
    GPIO.setup(port_num, GPIO.OUT)
def led_drive(reps, multiple, direction):
    for i in range(reps):
        for port_num in direction:                  # and direction
            GPIO.output(port_num, 1)
            sleep(0.11)
            if not multiple:
                GPIO.output(port_num, 0)
# сообщения в терминал
if board_type == "m":
    print "These are the connections for the Multiface LEDs test:"
    print "BUFFER DIRECTION SETTINGS, jumpers on all OUT positions (1-12)"
    print "GPIO 25 --- BUFFERS 1 \nGPIO 24 --- BUFFERS 2"
    print "GPIO 23 --- BUFFERS 3 \nGPIO 22 --- BUFFERS 4"
    print "GPIO 21 --- BUFFERS 5 \nGPIO 18 --- BUFFERS 6"
    print "GPIO 17 --- BUFFERS 7 \nGPIO 11 --- BUFFERS 8"
    print "GPIO 10 --- BUFFERS 9 \nGPIO 9 --- BUFFERS 10"
    print "GPIO 8 --- BUFFERS 11 \nGPIO 7 --- BUFFERS 12"

else:
    print "These are the connections for the Gertboard LEDs test:"
    print "jumpers in every out location (U3-out-B1, U3-out-B2, etc)"
    print "GP25 in J2 --- B1 in J3 \nGP24 in J2 --- B2 in J3"
    print "GP23 in J2 --- B3 in J3 \nGP22 in J2 --- B4 in J3"
    print "GP21 in J2 --- B5 in J3 \nGP18 in J2 --- B6 in J3"
    print "GP17 in J2 --- B7 in J3 \nGP11 in J2 --- B8 in J3"
    print "GP10 in J2 --- B9 in J3 \nGP9 in J2 --- B10 in J3"
    print "GP8 in J2 --- B11 in J3 \nGP7 in J2 --- B12 in J3"
    print "(If you don't have enough straps and jumpers you can install"
    print "just a few of them, then run again later with the next batch.)"

raw_input("When ready hit enter.\n")

try:
    led_drive(3, 0, ports)
    led_drive(1, 0, ports_rev)
    led_drive(1, 0, ports)
    led_drive(1, 0, ports_rev)
    led_drive(1, 1, ports)
    led_drive(1, 0, ports)
```

```
    led_drive(1, 1, ports)
    led_drive(1, 0, ports)
except KeyboardInterrupt:              # CTRL+C keyboard interrupt
    GPIO.cleanup()
```

Прежде всего проверяем версию платы Gertboard и определяем список портов `ports` для нашей ревизии. Затем настраиваем порты для вывода, руководствуясь списком `ports`. Функция `led_drive ()` управляет зажиганием-гашением светодиодов, для которой имеется три аргумента: `reps`, `multiple` и `direction`:

- `reps` определяет, сколь долго запускать процессы (1 или 3 в этой демонстрации);
- `multiple` определяет, возможно или нет выключение предыдущего перед включением следующего светодиода (значение 1 оставляет включенными несколько светодиодов);
- `direction` определяет направление, в котором следует листать список `ports` (`ports` — вперед, `ports_rev` — обратно).

В тексте программы содержится восемь вызовов функции `led_drive()`. Все они заключены в блоках `try` и `except`. Это гарантирует, что при выходе через комбинацию клавиш <Ctrl>+<C> значения портов GPIO будут сброшены (`cleanup`). Это позволяет избежать ложных включений, и порты смогут свободно использоваться другой программой.

Драйвер с открытым коллектором

Плата Gertboard располагает шестью портами, организованными на микросхеме ULN2803A и предоставляющими собой драйверы с открытым коллектором. Они служат, чтобы включать и выключать устройства, — в частности те, которые рассчитаны на питание от внешнего источника с различными напряжениями и токами большими, чем может позволить себе плата Gertboard. Схема подключения одного из шести блоков драйвера с открытым коллектором представлена на рис. 6.30.

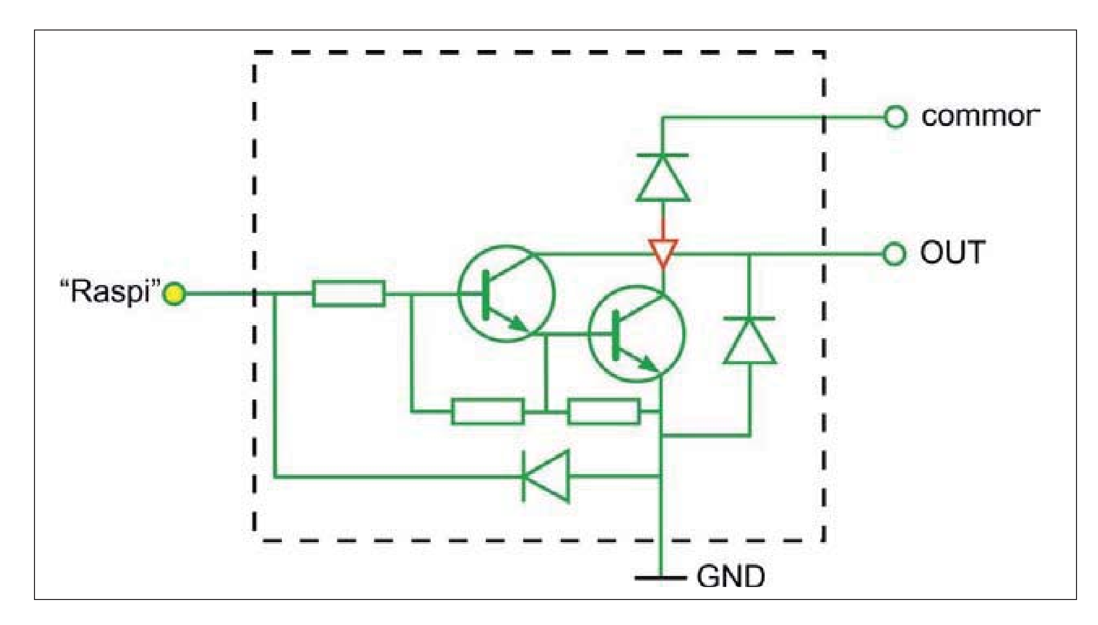


Рис. 6.30. Схема подключения блока драйвера с открытым коллектором

Для включения устройств драйвер с открытым коллектором замыкает цепи на плате на стороне внешних схем на "землю", таким образом давая им сигнал. Микросхема ULN2803A может выдержать напряжение до 50 В и ток до 500 мА на каждом из своих портов. Каждый порт драйвера имеет встроенный диод защиты.

Common — это провод цепи питания. Как следует из его названия, он питает сразу все блоки драйвера с открытым коллектором. Этот провод на плате Gertboard ни к чему не подключен. Как и все устройства управления, драйвер с открытым коллектором с помощью точки **Raspi** может быть подключен к портам GPIO или к выходам контроллера ATmega — например, для включения реле, двигателей либо иных управляемых устройств.

На рис. 6.28 и 6.29 контакты, соответствующие точкам **Raspi**, — это контакты с метками с RLY1 по RLY6 в разъеме J4. Контакты, соответствующие **common**, отмечены как RPWR в разъеме на правом краю платы, а контакты, соответствующие **OUT**, помечены RLY1 по RLY6 в разъемах с J12 по J17.

Контроллер двигателя

На борту платы Gertboard установлен контроллер двигателя, собранный на микросхеме ROHM BD6222HFP. Контроллер двигателя способен работать с коллекторными электродвигателями постоянного тока, рассчитанными на напряжение не более 18 В и ток не более 2 А. Контроллер имеет два контакта-входа A и B, обозначенные на плате как MOTA и MOTB (см. рис. 6.28 и 6.29). Контроллер двигателя управляется высоким и низким логическими уровнями по шаблону, приведенному в табл. 6.2. Скорость вращения двигателя контролируется путем применения сигнала широтно-импульсной модуляции (ШИМ).

Таблица 6.2. *Поведение контроллера в зависимости от сигналов на входах A и B*

A	B	Поведение двигателя
0	0	Нет движения
0	1	Вращение
1	0	Вращение в противоположную сторону
1	1	Нет движения

Контроллер двигателя находится в верхней части платы. Он имеет колодку с винтовыми зажимами-клеммами для крепления проводов двигателя и внешнего источника питания, а также два штырька-контакта для подачи сигналов управления в разъеме J5, находящемся чуть выше контактов GP4 и GP1 в разъеме J2. Штырьки-контакты с маркировкой MOTA и MOTB в разъеме J5 являются логическими входами низкого либо высокого уровня. Винтовые клеммы в верхней части платы, помеченные как MOTA и MOTB, представляют собой выходы контроллера двигателя. Для питания используется внешний источник, подключаемый с помощью винтовых клемм, защищенных плавким предохранителем F1 на 2 А. Предохранитель здесь требуется потому, что для работы двигателя необходимы ток и напряжение пита-

ния большие, чем может обеспечить плата Gertboard. Выход напряжения на винтовые клеммы MOTA и MOTB возможен только при наличии соответствующего модулируемого логического сигнала на штырьках-контактах разъема J5, помеченных как MOTA и MOTB.

Для включения двигателя в любом направлении без изменения скорости, необходимо выбрать пару свободных портов GPIO, установить электрическое соединение между ними и входами контроллера двигателя MOTA и MOTB и подать на них сигналы, соответствующие табл. 6.2.

С помощью ШИМ можно контролировать количество времени выхода высокого уровня по сравнению с временем низкого уровня. Выход импульсов широтно-импульсного модулятора, собранного на базе процессора BCM2835, установленного на борту Raspberry Pi, можно получить на выходе порта GPIO18, включив альтернативную функцию 5. Этот выход порта соединен с одним из входов контроллера двигателя — например, с входом MOTA, куда и подается прямоугольный импульс, при этом на вход MOTB подается постоянный высокий или низкий уровень. Таким образом и регулируются режимы работы двигателя.

Тестовые программы motor-rg.py и motor-wp.py существенно отличаются, потому что пакет RPi.GPIO пока еще не поддерживает аппаратный широтно-импульсный модулятор, тогда как WiringPi для Python поддерживает, хотя об этом прямо и не говорится. Соответственно, программа motor-rg.py в версии для RPi.GPIO основана на программно-организованном ШИМ.

Программа motor-rg.py (листинг 6.14) после импорта необходимых модулей определяет используемые порты GPIO 17 и 18, количество раз запуска каждого цикла — `Reps`, время рабочего цикла ШИМ — `Hertz` и частоту периодичности циклов двигателя — `Freq`. Затем порты настраиваются как выходы и устанавливаются в выключенное состояние — `OFF (0)`. Тем и определяется функция `run_motor(Reps, pulse_width, port_num, time_period)`, которая контролирует включение и выключение портов GPIO в определенный период времени: порт `port_num` включается на время `pulse_width`, затем отключается на время `time_period`. Все это повторяется `Reps` раз.

Основной цикл в функции `run_motor ()` находится внутри блока `try: except:`. Это важно для безопасности, поскольку обеспечивает правильное выключение двигателя прерыванием с клавиатуры. Функция `run_motor ()` вызывается множество раз функцией `run_loop()`. Каждый цикл в 400 повторений — это всего один шаг повышения или понижения скорости вращения двигателя. 400 повторений при частоте 2000 герц занимают лишь 0,2 секунды.

Рассмотрим параметры функции `run_loop(startloop, endloop, step, port_num, printchar)` подробнее:

- `startloop` и `endloop` выделяют время в % для коммутируемого порта и запускают либо останавливают цикл;
- `step` — размер приращения для каждого последующего цикла;
- `port_num` — порты, которые мы переключаем: 17 или 18;

- `printchar` — индикация ускорения/замедления с помощью количества знаков "+" или "–".

Как уже отмечено, функция `run_loop ()` вызывает функцию `run_motor ()`, заставляя ее повторяться 400 циклов для каждого набора определенного значения. После того как функции определены, инструкции соединений выведены, компьютер ждет нажатия клавиши <Enter>, подтверждающей, что пользователь готов. Затем функция `run_loop ()` вызывается четыре раза для управления скоростью двигателя от 5 до 95 % в каждом из направлений. После этого оба порта выключаются и сбрасываются.

Листинг 6.14. Программа motor-rg.py

```
from __future__ import print_function
import RPi.GPIO as GPIO
import sys
from time import sleep
board_type = sys.argv[-1]

GPIO.setmode(GPIO.BCM)
ports = [18,17]
Reps = 400
Freq = (1 / float(Hertz)) - 0.0003
for port_num in ports:
    GPIO.setup(port_num, GPIO.OUT)
    print ("setting up GPIO port:", port_num)
    GPIO.output(port_num, False)
def run_motor(Reps, pulse_width, port_num, time_period):
    try:
        for i in range(0, Reps):
            GPIO.output(port_num, True)
            sleep(pulse_width)
            GPIO.output(port_num, False)
            sleep(time_period)
    except KeyboardInterrupt:
        GPIO.cleanup()
def run_loop(startloop, endloop, step, port_num, printchar):
    for pulse_width_percent in range(startloop, endloop, step):
        print (printchar, sep='', end='')
        sys.stdout.flush()
        pulse_width = pulse_width_percent / float(100) * Freq
        time_period = Freq - (Freq * pulse_width_percent / float(100))
        run_motor(Reps, pulse_width, port_num, time_period)
    print("")

if board_type == "m":
    print ("\nThese are the connections for the Multiface motor test:")
    print ("GPIO 17 --- MOTB")
```

```
    print ("GPIO 18 --- MOTA")
    print ("+ of external power source --- MOTOR +")
    print ("ground of external power source --- MOTOR - ")
    print ("one wire for your motor in MOTOR A screw terminal")
    print ("the other wire for your motor in MOTOR B screw terminal")
else:
    print ("\nThese are the connections for the Gertboard motor test:")
    print ("GP17 in J2 --- MOTB (just above GP1)")
    print ("GP18 in J2 --- MOTA (just above GP4)")
    print ("+ of external power source --- MOT+ in J19")
    print ("ground of external power source --- GND (any)")
    print ("one wire for your motor in MOTA in J19")
    print ("the other wire for your motor in MOTB in J19")
command = raw_input("When ready hit enter.\n>")
print (">>>", sep='', end='')
run_loop(5, 95, 1, 18,'+')
run_loop(95, 5, -1, 18,'-')
sleep(0.2)
print ("<<<", sep='', end='')
run_loop(5, 95, 1, 17,'+')
run_loop(95, 5, -1, 17,'-')
GPIO.output(port_num, False)
GPIO.cleanup()
```

Теперь рассмотрим программу motor-wp.py (листинг 6.15). Raspberry Pi имеет один доступный аппаратный ШИМ-порт — GPIO18. Мы воспользуемся портом 17 — для управления направлением вращения двигателя и портом 18 — для выдачи прямоугольного импульса. Первая часть программы импортирует необходимые модули. После инициализации пакета WiringPi порт 18 устанавливается в режим ШИМ с помощью инструкции:

```
wiringpi.pinMode(18,2)
```

а порт 17 настраивается на нормальный выход. Затем оба порта устанавливаются в `0 (OFF)`. Далее инструкция подключений выводится на экран, и компьютер ждет команды пользователя. Когда пользователь готов, программа определяет три функции, которые используются повторно:

- `display` (`printchar`) — обрабатывает правильную индикацию ускорения/замедления двигателя;
- `reset_ports ()` — производит сброс портов при выходе из программы. Это важно для программы управления двигателем в пакете;
- `loop(start_pwm, stop_pwm, step, printchar)` — обслуживает основной цикл ШИМ.

В момент `start_pwm` — ШИМ принимает значения от 0 до 1024 в начале цикла. В момент `stop_pwm` — ШИМ принимает значение от 0 до 1024 в конце цикла. Параметр `step` — приращивает/уменьшает значение для каждого последующего цикла.

Параметр `printchar` — оператор для отображения режима ускорения или замедления вращения двигателя. Основная часть программы содержит четыре вызова `loop()`, чтобы продемонстрировать ускорение и замедление в каждом из двух различных направлений вращения с короткими, но ощутимыми, паузами между циклами. Время паузы определяется в `rest = 0,013` (0,013 секунды). Когда значение порта 17 равно 0, двигатель вращается в одном направлении. Когда значение порта 17 равно 1, он вращается в противоположную сторону. Как обычно, в основной части программы содержится конструкция `try: except:`, предусматривающая включение блокировки и сброса значения порта для обеспечения безопасности перед выходом с применением прерывания с клавиатуры.

Листинг 6.15. Программа motor-wp.py

```
from __future__ import print_function
import wiringpi
import sys
from time import sleep
board_type = sys.argv[-1]

wiringpi.wiringPiSetupGpio()
wiringpi.pinMode(18,2)
wiringpi.pinMode(17,1)
wiringpi.digitalWrite(17, 0)
wiringpi.pwmWrite(18,0)
if board_type == "m":
    print ("\nThese are the connections for the Multiface motor test:")
    print ("GPIO 17 --- MOTB")
    print ("GPIO 18 --- MOTA")
    print ("+ of external power source --- MOTOR +")
    print ("ground of external power source --- MOTOR - ")
    print ("one wire for your motor in MOTOR A screw terminal")
    print ("the other wire for your motor in MOTOR B screw terminal")
else:
    print ("\nThese are the connections for the Gertboard motor test:")
    print ("GP17 in J2 --- MOTB (just above GP1)")
    print ("GP18 in J2 --- MOTA (just above GP4)")
    print ("+ of external power source --- MOT+ in J19")
    print ("ground of external power source --- GND (any)")
    print ("one wire for your motor in MOTA in J19")
    print ("the other wire for your motor in MOTB in J19")
command = raw_input("When ready hit enter.\n>")
def display(printchar):
    print (printchar, sep='', end='')
    sys.stdout.flush()
def reset_ports():
    wiringpi.pwmWrite(18,0)
    wiringpi.digitalWrite(18, 0)
```

```
    wiringpi.digitalWrite(17, 0)
    wiringpi.pinMode(17,0)
    wiringpi.pinMode(18,0)
def loop(start_pwm, stop_pwm, step, printchar):
    for x in range(start_pwm, stop_pwm, step):
        wiringpi.pwmWrite(18,x)
        # if x is an exact multiple of 19, i.e. x/19 has remainder 0
        if x % (19) == 0:
            display(printchar)
        sleep(rest)
rest = 0.013
try:
    display('>>> ')
    loop(140, 1024, 1, '+')
    loop(994, 110, -1, '-')
    wiringpi.digitalWrite(17, 1)
    display('\n<<< ')
    loop(954, 89, -1, '+')
    loop(121, 1024, 1, '-')
    display('\n')
except KeyboardInterrupt:
    reset_ports()
reset_ports()
```

Аналого-цифровые и цифроаналоговые преобразователи

На плате Gertboard установлен цифроаналоговый преобразователь (D/A) от компании Microchip, собранный на микросхеме MCP48*x*2. Это устройство производится в трех различных вариантах: 8-, 10- или 12-битном (разрядном). Какой из вариантов установлен именно на вашей плате, можно узнать из маркировки на корпусе микросхемы. Для этого посмотрите очень внимательно на маленький чип, обозначенный U10. На его корпусе должен быть виден номер 48*x*2, где вместо *x* могут стоять числа: 0, 1 или 2. Если *x* равен 0, то преобразователь 8-разрядный, если 1 — то 10-разрядный, а если 2 — то 12-разрядный. Все эти варианты исполнения микросхемы совместимы и имеют одинаковое расположение и счет выводов. Программа, которая работает с D/A, предполагает, что записывает в 12-битный чип, поэтому важно указывать соответствующие значения. Максимальные выходное и входное напряжения D/A — 2,04 В. Аналоговые выходы двух каналов подключены к контактам, помеченным как DA0 для канала 0 и DA1 — для канала 1 в разъеме J29 на левом краю платы. Контакты, расположенные рядом, подключены к "земле" (GND).

На плате Gertboard используется 10-битный аналого-цифровой преобразователь (A/D), собранный на микросхеме MCP3002. Он поддерживает 2 канала с частотой дискретизации около 72K операций в секунду (sps). Максимальное значение — 1023 — возможно, когда входное напряжение равно 3,3 В. Аналоговые входы двух каналов AD0 и AD1 находятся в разъеме J28. Рядом с этими выводами расположены контакты "земли" (GND).

Тестирование D/A и A/D

Как сказано в руководстве на микросхему MCP4802, значение напряжения (В) на выходе V_{out} вычисляется по формуле:

$$V_{out} = (Din/2^x) \times 2{,}048,$$

где $x = 8$, 10, 12 — разрядность микросхемы MCP4802.

Выход V_{out} для канала 0 представлен контактом DA0 в разъеме J29, а для канала 1 — DA1.

Для проверки работы цифроаналогового преобразователя (D/A) потребуется вольтметр или мультиметр. Программа тестирования для Python называется dtoa.py. Чтобы настроить плату Gertboard для этого теста, необходимо установить перемычки на контакты портов GP11, GP10, GP9, GP7 и на контакты шины SPI, находящиеся над ними.

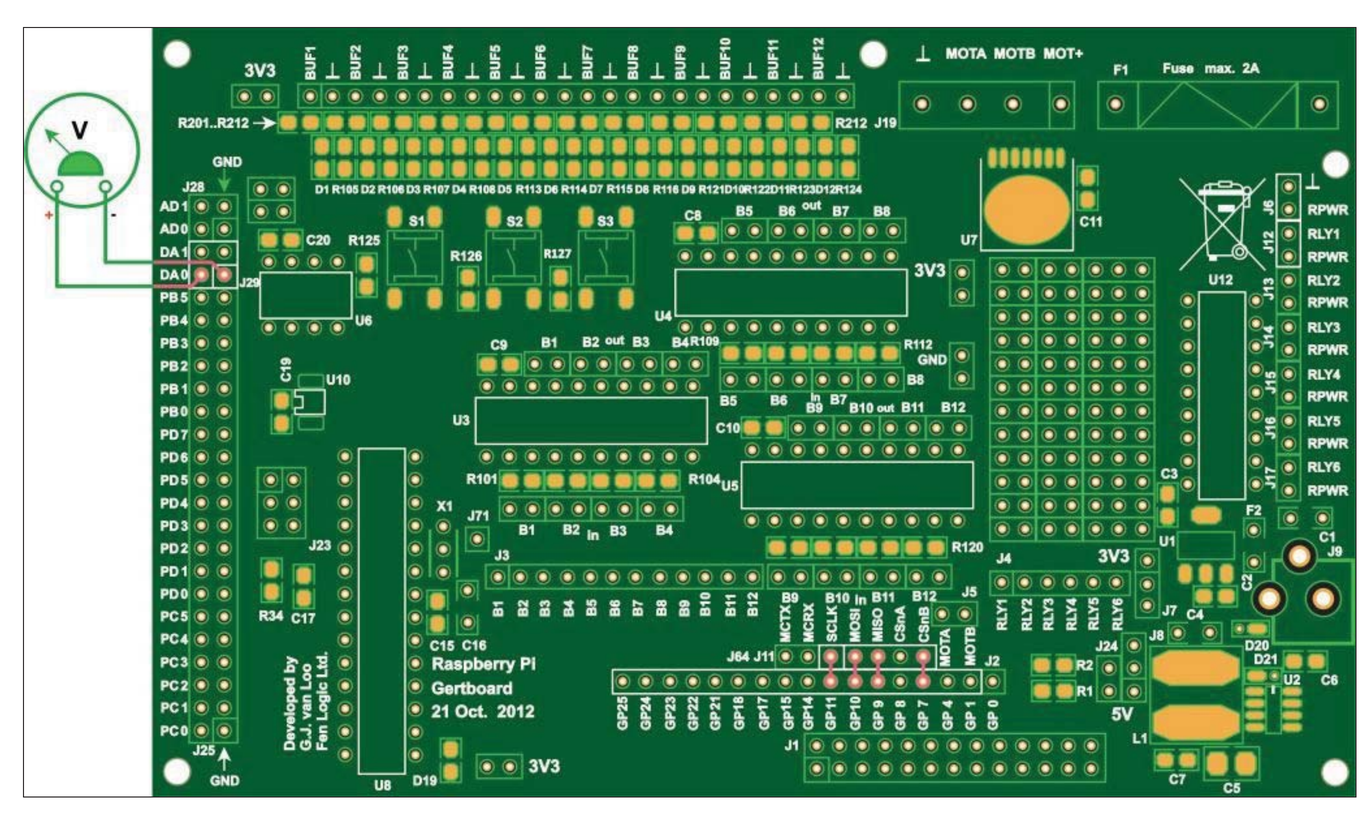


Рис. 6.31. Схема соединений для теста dtoa.py

Подключить мультиметр надо следующим образом: черный отрицательный провод должен быть подсоединен к GND. Для этого можно использовать любой из штырьков, помеченных как GND. Красный, положительный, провод должен быть подсоединен к DA0 (для проверки D/A канала 0) или DA1 (для проверки D/A канала 1). Переключите мультиметр и установите его для измерения постоянного напряжения. Схема соединений показана на рис. 6.31.

Содержимое файла dtoa.py представлено в листинге 6.16. Цифроаналоговый преобразователь (D/A) управляется с помощью записи двоичных чисел (16 битов в целом) через шину интерфейса SPI. Программа для модуля Python называется `spidev` и использует шину SPI для связи. Мы должны дать `spidev` двоичные 10-битные

значения, которые он преобразует в 8-битный двоичный код, впоследствии передаваемый в D/A. D/A выдает напряжение в соответствии с цифровым значением на входе. Если на входе цифровое значение 0, то это соответствует 0 В, а если значение 255, то на выходе 2,048 В, и это линейная зависимость между ними, так что цифровое значение 128 должно соответствовать напряжению 1,02 В.

Мы посылаем в D/A заранее определенные значения, которые хранятся в двух изменяемых списках:

- `num_list []` — содержит первый байт данных (отличается для каждого канала);
- `common []` — содержит второй байт данных (одинаковые для каждого канала).

Пользователь выбирает канал, а затем основной цикл принимает первый номер из каждого списка и сливает `byte1` с `byte2`. Затем он посылает их в D/A, который сразу же выдает соответствующее напряжение на выходе до нажатия клавиши <Enter>. Цикл проходит по всем пяти значениям, установленным в D/A, и дожидается нажатия клавиши <Enter>. Оба канала сбрасываются в ноль в конце программы.

Листинг 6.16. Программа dtoa.py

```
import spidev
import sys
from time import sleep
board_type = sys.argv[-1]

import subprocess
unload_spi = subprocess.Popen('sudo rmmod spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
start_spi = subprocess.Popen('sudo modprobe spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
sleep(3)

def which_channel():
    channel = raw_input("Which channel do you want to test? Type 0 or 1.\n")
    while not channel.isdigit():
        channel = raw_input("Try again - just numbers 0 or 1 please!\n")      #
Make them do it again if wrong
    return channel

spi = spidev.SpiDev()
spi.open(0,1)

channel = 3
common = [0,0,0,160,240]
voltages = [0.0,0.5,1.02,1.36,2.04]

while not (channel == 1 or channel == 0):
    channel = int(which_channel())
```

```
if channel == 1:
    num_list = [176,180,184,186,191]
else:
    num_list = [48,52,56,58,63]

print "These are the connections for the digital to analogue test:"

if board_type == "m":
    print "jumper connecting GPIO 7 to CSB"
    print "Multimeter connections (set your meter to read V DC):"
    print "  connect black probe to GND"
    print "  connect red probe to DA%d on D/A header" % channel

else:
    print "jumper connecting GP11 to SCLK"
    print "jumper connecting GP10 to MOSI"
    print "jumper connecting GP9 to MISO"
    print "jumper connecting GP7 to CSnB"
    print "Multimeter connections (set your meter to read V DC):"
    print "  connect black probe to GND"
    print "  connect red probe to DA%d on J29" % channel

raw_input("When ready hit enter.\n")

for i in range(5):
    r = spi.xfer2([num_list[i],common[i]])
    print "Your meter should read about %.2fV" % voltages[i]
    raw_input("When ready hit enter.\n")

r = spi.xfer2([16,0])  # switch off channel A = 00010000 00000000 [16,0]
r = spi.xfer2([144,0]) # switch off channel B = 10010000 00000000 [144,0]
```

Программа для теста аналого-цифрового преобразователя называется atod.py. Чтобы запустить этот тест, источника напряжения на аналоговом входе не требуется. Проще всего использовать потенциометр (переменный резистор). Два конца потенциометра связаны с одной стороны с высоким уровнем 3,3 В (который можно получить из любого контактного разъема с отметкой 3V3), а с другой стороны с низким уровнем, или GND. Средний конец связан со входом AD0 (для канала 0) или AD1 (для канала 1). Для использования шины SPI соединительные провода должны быть установлены на контакты GP11, GP10, GP9, GP8 и подключены к контактам шины SPI, которые находятся над ними, как показано на рис. 6.32.

Содержимое файла atod.py представлено в листинге 6.17. Пользователь выбирает канал, который будет использоваться аналого-цифровым преобразователем (A/D), как и написано в комментариях к программе. Функция `get_adc (channel)` использует программу `spidev` для считывания A/D. Эта функция получает три байта данных

и извлекает результат — число от 0 до 1023, где 0 означает 0 В, а 1023 — 3,3 В. Главный цикл выполняется 600 раз с задержкой `sleep` в 0,05 с, поэтому выполнение программы занимает около 30 секунд: 600×0,05. Во время каждой итерации цикла программа считывает значение из A/D и отображает результат количеством символов "#" пропорционально считанному значению, которое зависит от положения движка потенциометра. Для запуска этого процесса служит функция `display(char, reps, adc_value, spaces)`. Во время работы программы перемещение движка потенциометра приводит к изменениям как цифровых величин, так и количества отображаемых на дисплее символов "#".

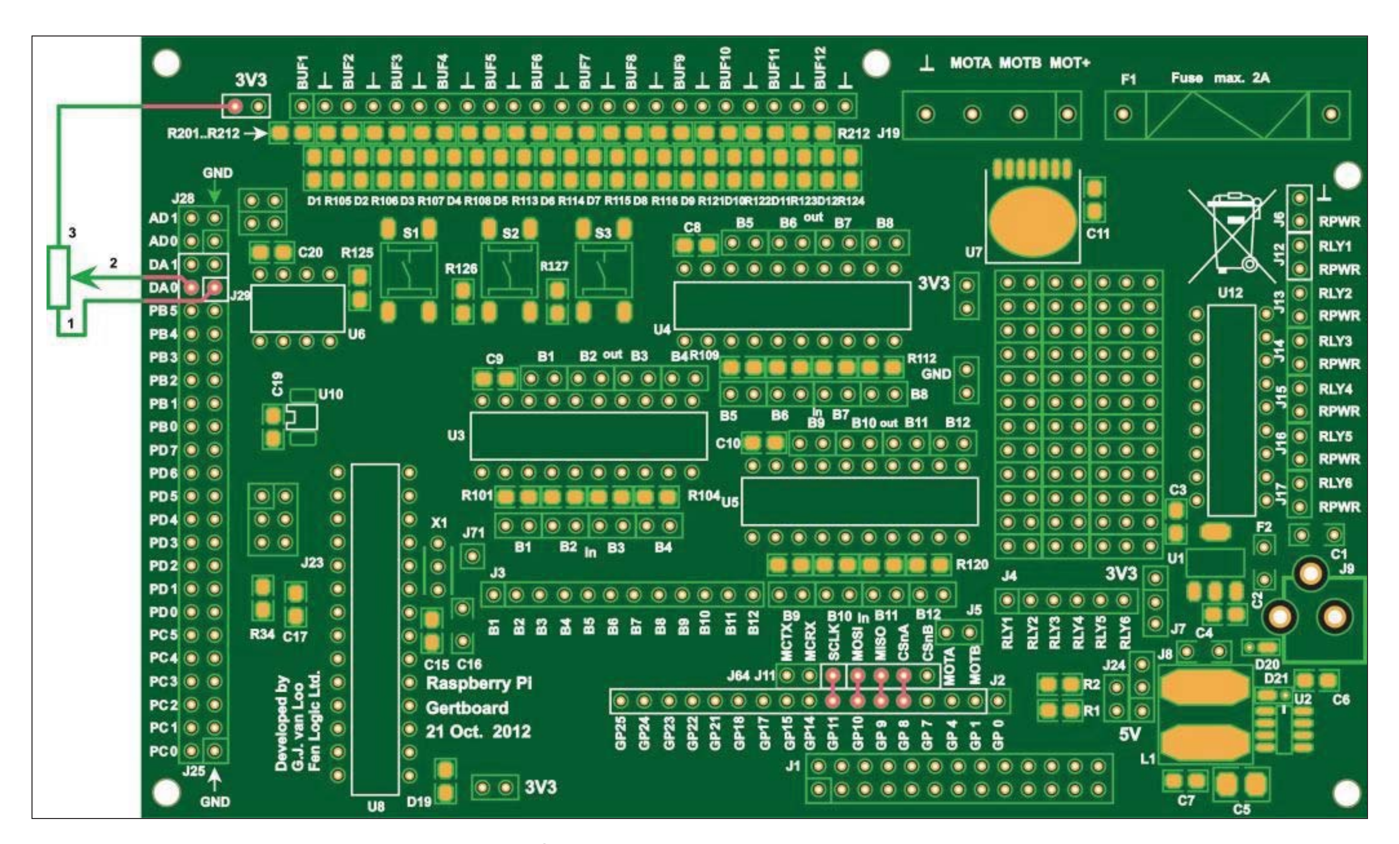


Рис. 6.32. Схема соединений для теста atod.py

Листинг 6.17. Программа atod.py

```
from __future__ import print_function
from time import sleep

import subprocess
unload_spi = subprocess.Popen('sudo rmmod spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
start_spi = subprocess.Popen('sudo modprobe spi_bcm2708', shell=True,
stdout=subprocess.PIPE)
sleep(3)

import spidev
import sys
board_type = sys.argv[-1]
```

```
def which_channel():
    channel = raw_input("Which channel do you want to test? Type 0 or 1.\n")
    while not channel.isdigit():
        channel = raw_input("Try again - just numbers 0 or 1 please!\n")
    return channel

def get_adc(channel):
if ((channel > 1) or (channel < 0)):
else return -1
        return -1
    r = spi.xfer2([1,(2+channel)<<6,0])
    ret = ((r[1]&31) << 6) + (r[2] >> 2)
    return ret

def display(char, reps, adc_value, spaces):            print
('\r',"{0:04d}".format(adc_value), ' ', char * reps, ' ' * spaces,'\r', sep='',
end='')
    sys.stdout.flush()

iterations = 0
char = '#'
channel = 3
while not (channel == 1 or channel == 0):
    channel = int(which_channel())
    print ("These are the connections for the analogue to digital test:")

if board_type == "m":
    print ("jumper connecting GPIO 8 to CSA")

else:
    print ("jumper connecting GP11 to SCLK")
    print ("jumper connecting GP10 to MOSI")
    print ("jumper connecting GP9 to MISO")
    print ("jumper connecting GP8 to CSnA")

print ("Potentiometer connections:")
print ("  (call 1 and 3 the ends of the resistor and 2 the wiper)")
print ("  connect 3 to 3V3")
print ("  connect 2 to AD%d" % channel);
print ("  connect 1 to GND")

raw_input("When ready hit enter.\n")

spi = spidev.SpiDev()
spi.open(0,0)
while iterations < 600:
    adc_value = (get_adc(channel))
```

```
    reps = adc_value / 16
    spaces = 64 - reps
    display(char, reps, adc_value, spaces)
    sleep(0.05)
    iterations += 1
```

ПРИМЕЧАНИЕ

Коды скриптов этого раздела вы найдете в папке *glava_06\GB_Python* сопровождающего книгу электронного архива (см. *приложение*).

6.3.4. Программирование ATmega

Программирование микроконтроллеров ATmega просто, если у вас настроена вся инфраструктура и на борту микрокомпьютера Raspberry Pi установлено специальное программное обеспечение. Для программирования можно использовать плату Arduino IDE. Чтобы соединить Arduino IDE с Raspberry Pi и Gertboard, нужно будет сделать несколько небольших изменений: установить дополнительные пакеты и осуществить правку файлов конфигурации Arduino. Надо отметить, что эта работа была проведена Гордоном Хендерсоном (см. *разд. 6.1.3*).

Итак, сначала устанавливаем Arduino IDE:

```
sudo apt-get install arduino
```

Теперь загрузим и установим модифицированный пакет `avrdude`:

```
cd /tmp
wget http://project-downloads.drogon.net/gertboard/avrdude_5.10-4_armhf.deb
sudo dpkg -i avrdude_5.10-4_armhf.deb
sudo chmod 4755 /usr/bin/avrdude
```

Далее скачаем и выполним сценарий для изменения некоторых системных файлов:

```
cd /tmp
wget http://project-downloads.drogon.net/gertboard/setup.sh
chmod +x setup.sh
sudo ./setup.sh
```

Следующий шаг — сообщить Arduino IDE о процессоре ATmega на нашей плате Gertboard. Для этого подменяем в Arduino IDE файлы boards.txt и programmers.txt, добавив в них соответствующую информацию:

```
cd /tmp
wget http://project-downloads.drogon.net/gertboard/boards.txt
wget http://project-downloads.drogon.net/gertboard/programmers.txt
cd /usr/share/arduino/hardware/arduino
sudo mv boards.txt board.txt.bak
sudo mv /tmp/boards.txt .
sudo mv programmers.txt programmers.txt.bak
sudo mv /tmp/programmers.txt .
```

Прежде чем начать программировать ATmega, следует инициализировать чип. Но сначала надо подключить Gertboard к Raspberry Pi и установить четыре перемычки от GPIO Raspberry Pi к порту программатора ATmega ISP следующим образом (рис. 6.33):

- GPIO контакт 8 → ISP контакт 5 (сброс);
- GPIO контакт 9 → ISP контакт 1 (MISO);
- GPIO контакт 10 → ISP контакт 4 (MOSI);
- GPIO контакт 11 → ISP контакт 3 (SCLK).

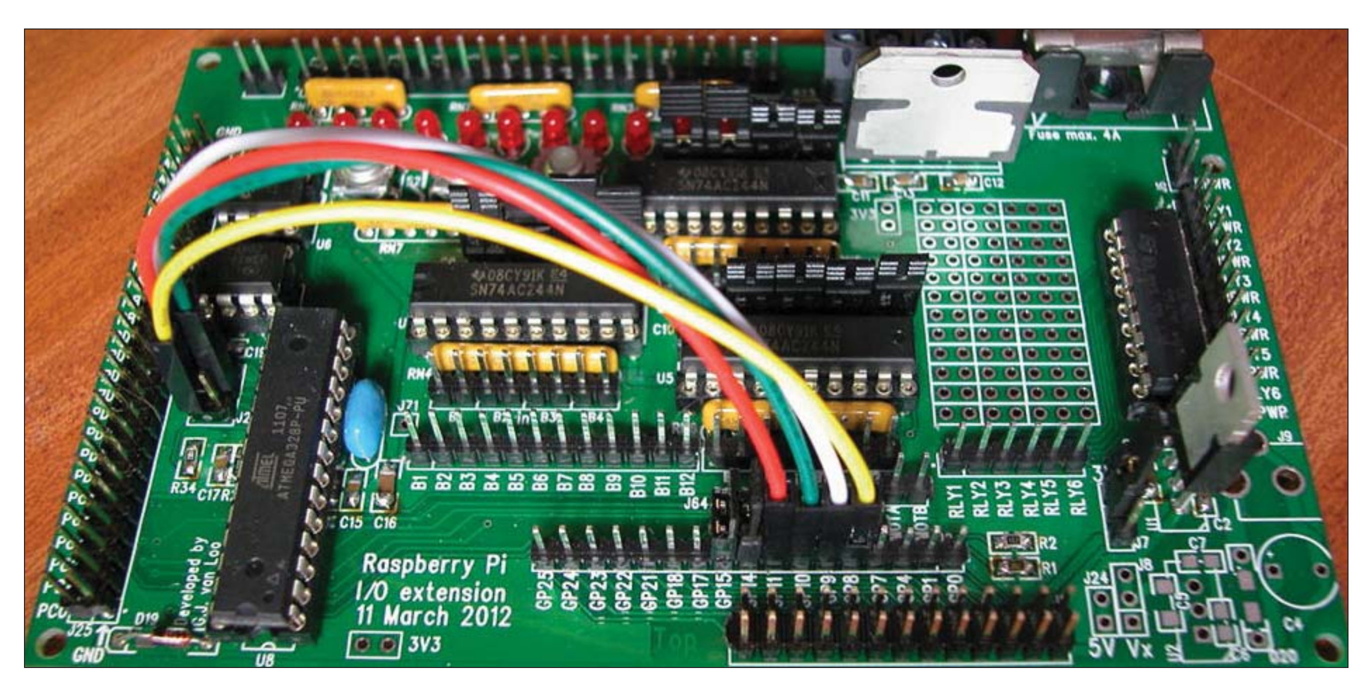


Рис. 6.33. Перемычки для соединения ATmega ISP

Теперь запустим скрипт для инициализации процессора ATmega:

```
avrsetup
```

Если все сделано правильно, вы должны увидеть результат, подобный изображенному на рис. 6.34.

На следующем шаге нужно настроить среду IDE для работы с платой Gertboard. Перейдите к настройке: **Tool | Board** и выберите опцию **Gertboard** с тем типом

```
pi@raspberrypi ~ $ cd /usr/share/arduino/hardware/arduino
pi@raspberrypi /usr/share/arduino/hardware/arduino $ avrsetup

Initialising a new ATmega microcontroller for use with the Gertboard.

Make sure there is a new ATmega chip plugged in, and press
.. 1 for an ATmega328p or 2 for an ATmega168: 1
Initialising an ATmega328p ...
Looks all OK - Happy ATmega programming!
pi@raspberrypi /usr/share/arduino/hardware/arduino $
```

Рис. 6.34. Выбор платы ATmega при выполнении скрипта `avrsetup`

микросхемы, который установлен на плате, — это могут быть ATmega168 или ATmega328, они чаще всего используются совместно с платой Gertboard (рис. 6.35). Затем перейдите в меню **Tools | Programmer** и выберите **Raspberry Pi GPIO** (рис. 6.36).

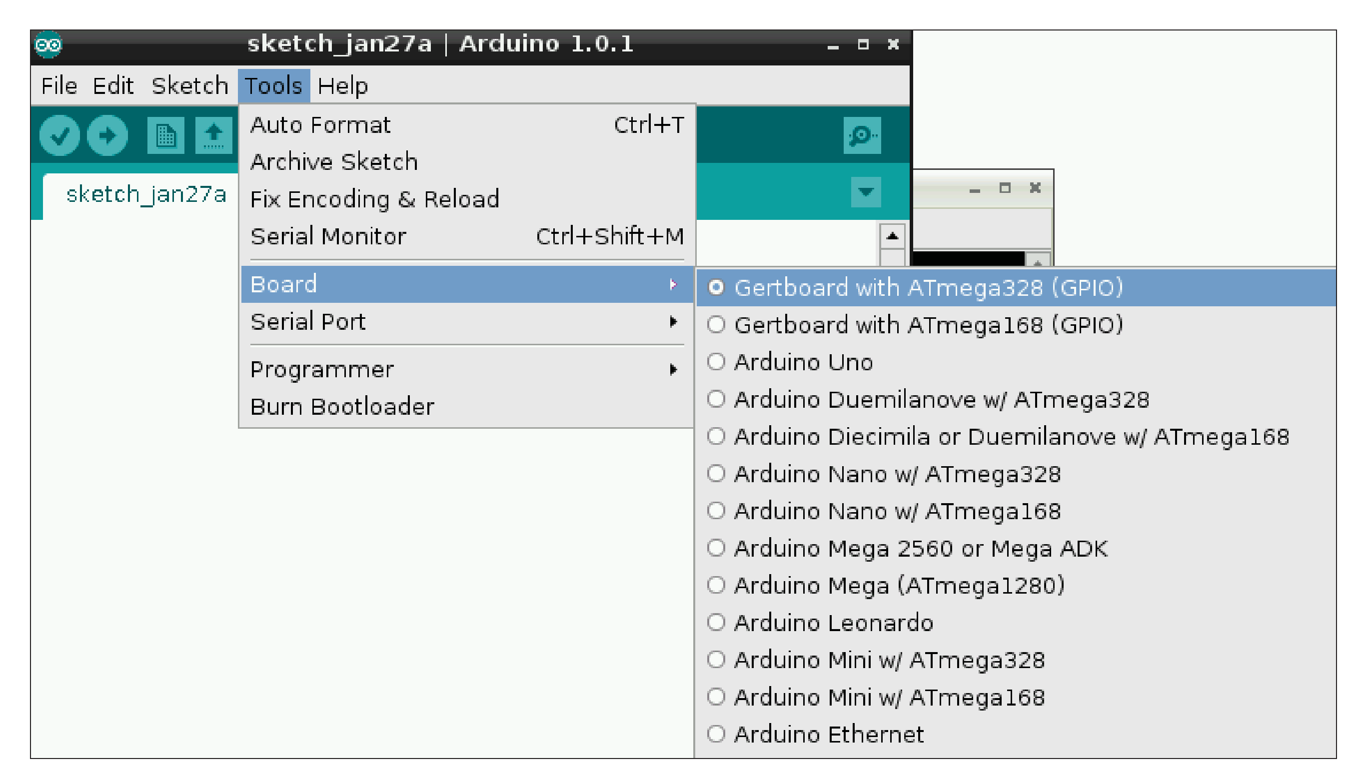


Рис. 6.35. Настройка платы Gertboard в Arduino IDE

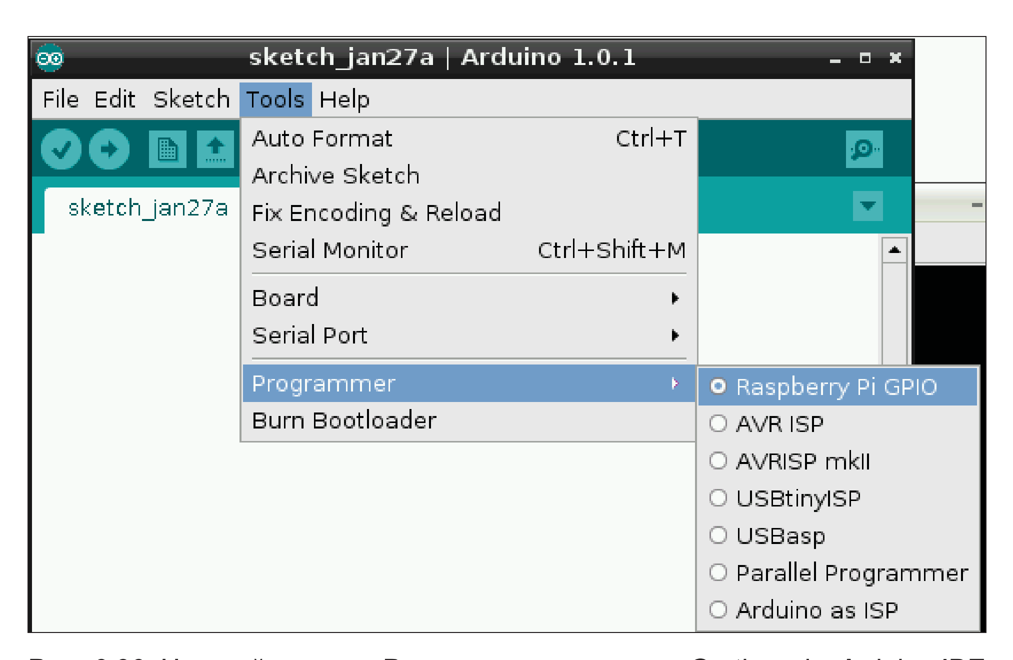


Рис. 6.36. Настройка опции **Programmer** для платы Gertboard в Arduino IDE

Все входные и выходные выводы микросхемы ATmega расположены в разъеме J25, в левом краю платы. Они помечены как PC*n*, PD*n* и PB*n*, где *n* — порядковый номер. Эти метки соответствуют "распиновке" чипов ATmega168/328. Тем не менее,

в мире Arduino номера контактов чипов не соответствуют номерам их функций. Вместо этого есть некоторые абстрактные обозначения цифровых и аналоговых выводов, которые не зависят от номеров выводов самих физических устройств. Это позволяет коду, написанному для одного устройства Arduino, успешно работать на другом устройстве Arduino, которое может иметь чип с другой раскладкой выводов. В табл. 6.3 показано соответствие выводов платы Gertboard с Arduino IDE.

Таблица 6.3. *Соответствие выводов платы Gertboard и Arduino*

Контакты Arduino	Контакты платы Gertboard
0	PD0
1	PD1
2	PD2
3	PD3
4	PD4
5	PD5
6	PD6
7	PD7
8	PB0
9	PB1
10	PB2
11	PB3
12	PB4
13	PB5
A0	PC0
A1	PC1
A2	PC2
A3	PC3
A4	PC4
A5	PC5

Теперь можно приступить к написанию скетча (программы для Arduino). Примем за основу программу **Blink** из примеров к Arduino IDE (**File | Examples | Basics | Blink**). Чтобы загрузить скетч в чип Arduino IDE, выполните команду меню **File | Upload Using Programmer** (рис. 6.37).

Это займет некоторое время, необходимое для компиляции и загрузки, а затем ваш скетч заработает на микроконтроллере. Чтобы наглядно увидеть работу скетча (изменение состояния на входе PB5 разъема J25 с периодичностью 1 сек.), подсоединим светодиод 1 перемычкой с PB5 на BUF1, и он станет мигать.

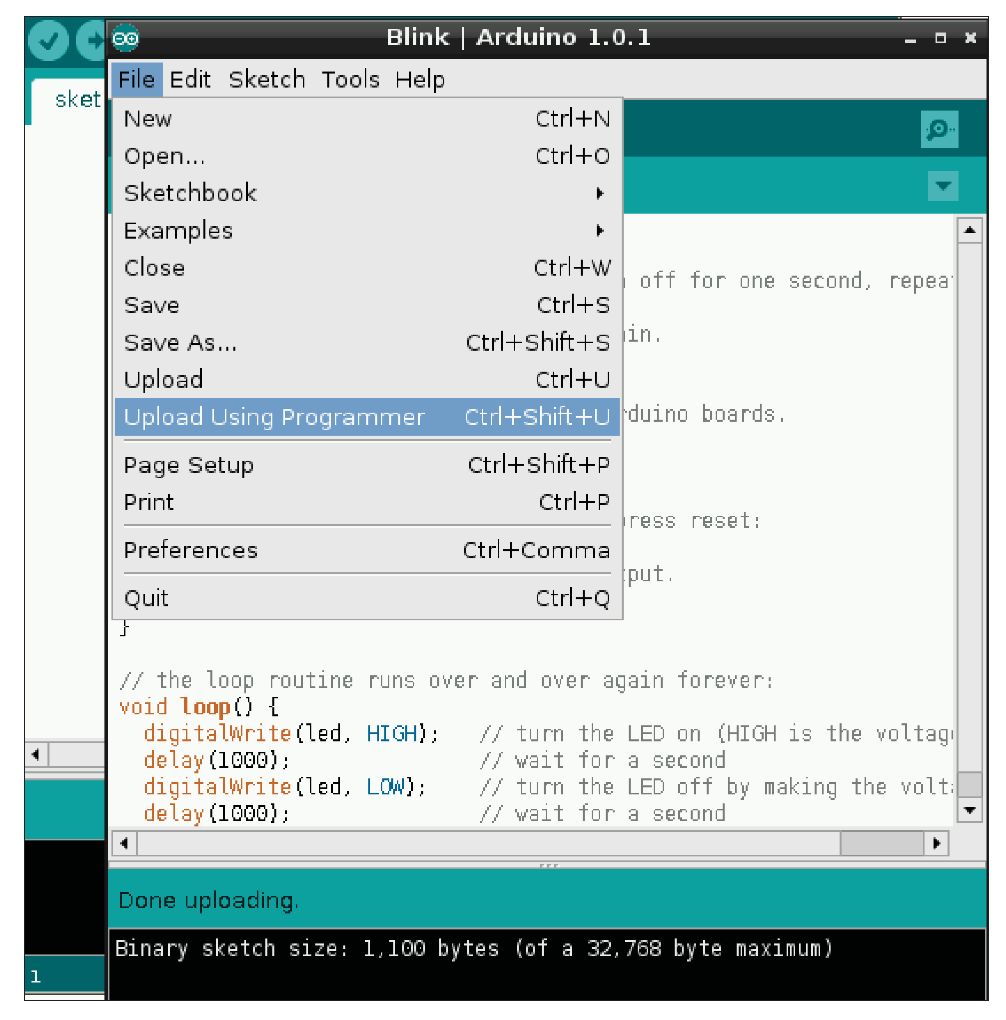


Рис. 6.37. Загрузка скетча в ATmega на Gertboard

Иногда при загрузке скетча может возникнуть ошибка (рис. 6.38). Чаще всего это происходит по двум причинам:

- загрузка осуществляется командой **Upload** (т. е. **File | Upload**), а правильно применить команду **Upload Using Programmer** (**File | Upload Using Programmer**);
- нет перемычки для подключения питания 3,3 В (2 верхних контакта на разъеме J7).

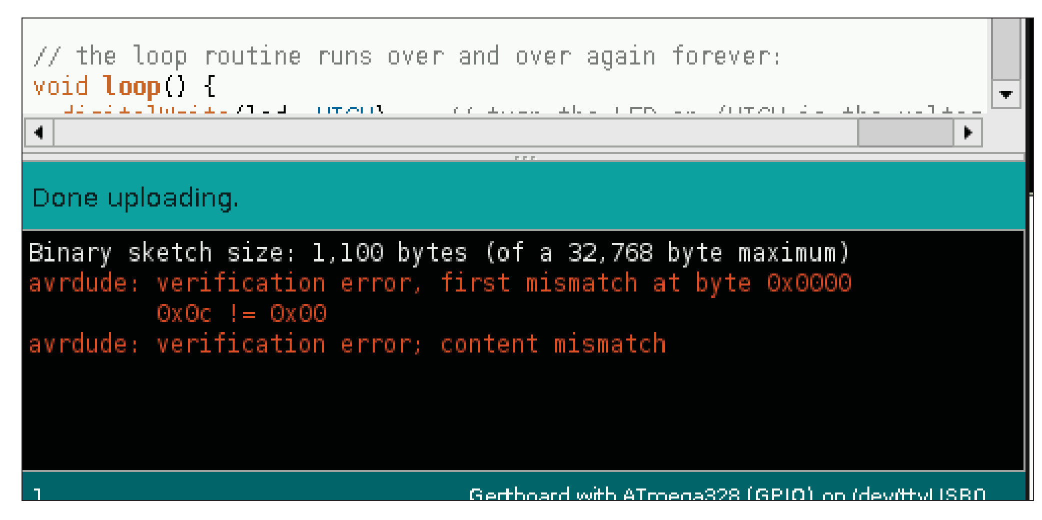


Рис. 6.38. Ошибка загрузки скетча в ATmega на Gertboard

По умолчанию Gertboard использует для GPIO последовательный порт AMA0. Однако Arduino IDE не воспринимает устройство /dev/ttyAMA0 как последовательный порт, поэтому мы переназначим порт на устройство /dev/ttyS1:

```
sudo ln /dev/ttyAMA0 /dev/ttyS1
```

Связывание делаем постоянным:

```
sudo nano /etc/init.d/link_serial
```

Заносим в файл `link_serial` следующий код:

```
#!/bin/bash
ln -s /dev/ttyAMA0 /dev/ttyS1
```

И далее:

```
sudo chmod 755 /etc/init.d/link_serial
update-rc.d link_serial defaults
```

Теперь при запуске Arduino IDE появится порт /dev/ttyS1 (рис. 6.39).

Проверяем, так ли это. Пишем небольшой скетч (листинг 6.18), загружаем его в Gertboard на ATmega, открываем монитор последовательного порта — и видим вывод данных скетча в последовательный порт (рис. 6.40).

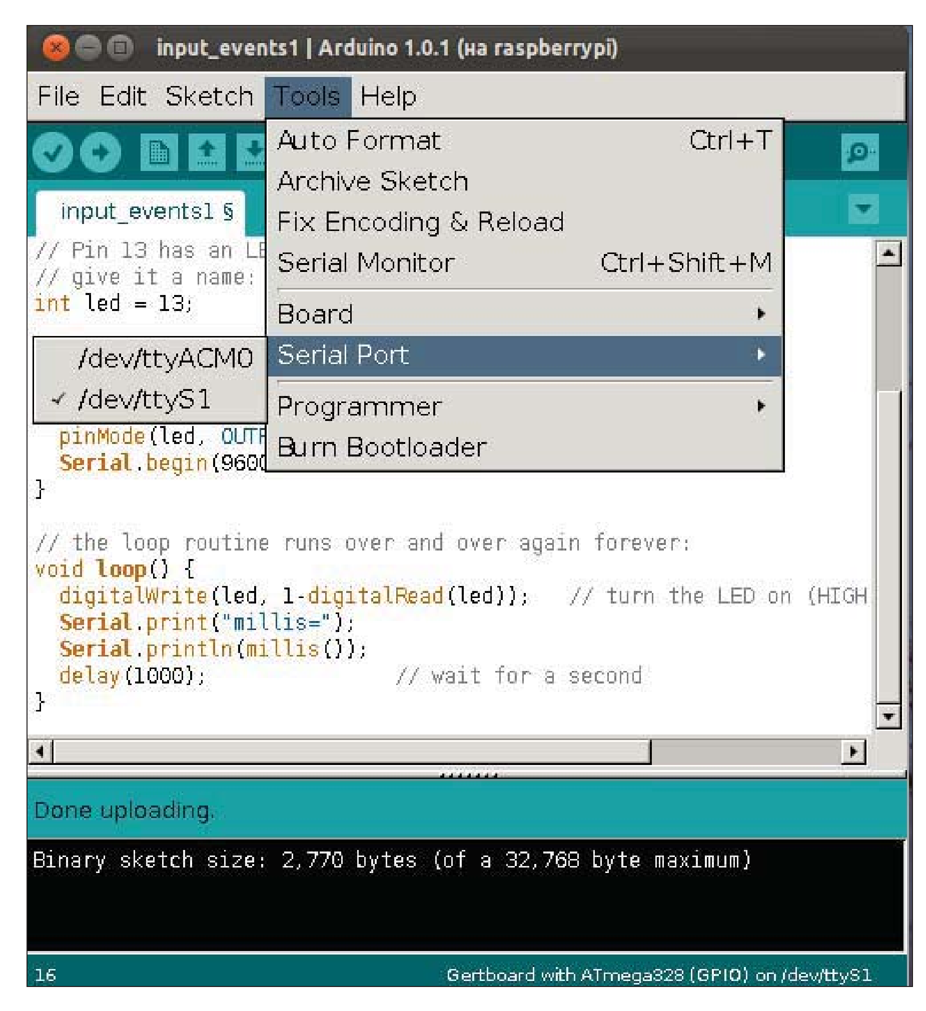


Рис. 6.39. Порт /dev/ttyS1 в Arduino IDE

Листинг 6.18. Скетч для вывода данных в последовательный порт

```
void setup()
   {
   Serial.begin(9600);
   }
void loop()
  {
  Serial.print("millis=");
  Serial.println(millis());
  delay(1000);
  }
```

```
/dev/ttyS1 (на raspberrypi)

millis=243430
millis=244494
millis=245557
millis=246619
millis=247683
millis=248746
millis=249808
millis=250872
millis=251935
millis=252998
millis=254061
millis=255125
millis=256187
```

Рис. 6.40. Просмотр данных в мониторе последовательного порта

6.3.5. Проект для платы Gertboard: контроль входа

Рассмотрим небольшой проект применения платы Gertboard в связке с Raspberry Pi — пропускной пункт со входом по брелокам или картам с RFID-меткой (рис. 6.41).



Рис. 6.41. Брелоки и карты RFID

RFID-считыватель (рис. 6.42) установлен на входной двери и подсоединен к контроллеру ATmega платы Gertboard. Датчик считывания RFID-карт компании Seeed Technology Inc., работающий на частоте 125 кГц, имеет высокую чувствительность расстояния срабатывания — 7 см. Он выдает информацию о карте в двух форматах данных: Uart и Wiegand. Wiegand — простой проводной интерфейс связи между устройством чтения идентификатора (карточки) и контроллером, широко применяемый в системах контроля доступа. Он как раз и предназначен для передачи уникального кода карты в контроллер.

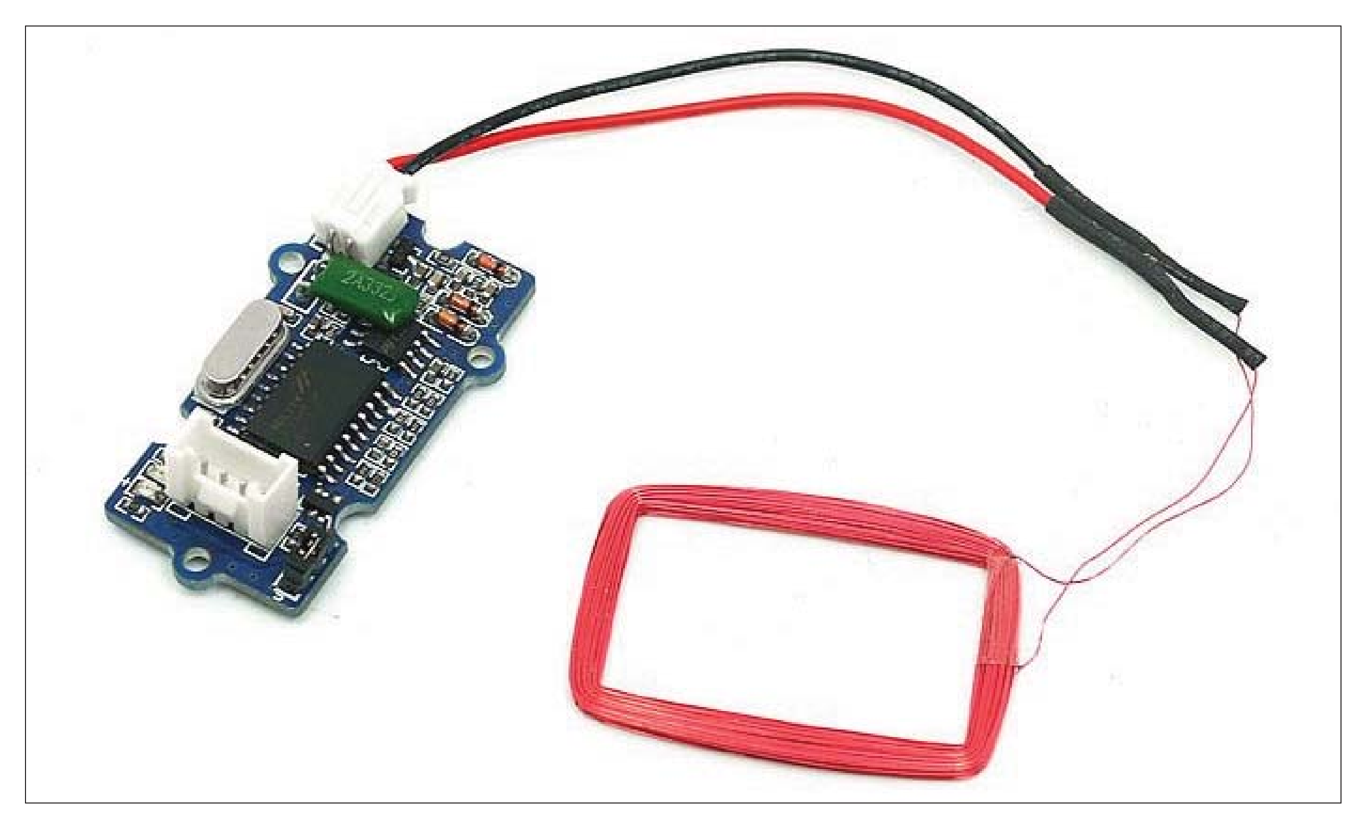


Рис. 6.42. RFID-считыватель

Программа для RFID-считывателя на ATmega

Схема подключения считывателя представлена на рис. 6.43.

RFID-считыватель определяет код поднесенного брелока или карты и отправляет по последовательному порту в Raspberry Pi. Содержимое скетча для получения кода из RFID-считывателя и отправки в Raspberry Pi по последовательному порту представлено в листинге 6.19.

Листинг 6.19. Скетч для получения кода из RFID-считывателя и отправки в Raspberry Pi

```
byte RFIDcardNum[4];
byte evenBit = 0;
byte oddBit = 0;
byte isData0Low = 0;
```

```
byte isData1Low = 0;
int recvBitCount = 0;
byte isCardReadOver = 0;

void setup()
{
 Serial.begin(9600);
 attachInterrupt(0, ISRreceiveData0, FALLING );
 attachInterrupt(1, ISRreceiveData1, FALLING );
}
void loop()
{
 // чтение бита номера карты
 if(isData0Low||isData1Low){
  if(1 == recvBitCount){//even bit
   evenBit = (1-isData0Low)&isData1Low;
  }
  else if( recvBitCount >= 26){//odd bit
   oddBit = (1-isData0Low)&isData1Low;
   isCardReadOver = 1;
   delay(10);
  }
  else{
   //only if isData1Low = 1, card bit could be 1
   RFIDcardNum[2-(recvBitCount-2)/8] |= (isData1Low << (7-(recvBitCount-2)%8));
  }
  //reset data0 and data1
  isData0Low = 0;
  isData1Low = 0;
 }
 // печать метки карты
 if(isCardReadOver){
  if(checkParity()){
   Serial.println(*((long *)RFIDcardNum));
  }
  resetData();
 }
}
byte checkParity(){
 int i = 0;
 int evenCount = 0;
 int oddCount = 0;
 for(i = 0; i < 8; i++){
  if(RFIDcardNum[2]&(0x80>>i)){
   evenCount++;
  }
 }
```

```
 for(i = 0; i < 4; i++){
  if(RFIDcardNum[1]&(0x80>>i)){
   evenCount++;
  }
 }
 for(i = 4; i < 8; i++){
  if(RFIDcardNum[1]&(0x80>>i)){
   oddCount++;
  }
 }
 for(i = 0; i < 8; i++){
  if(RFIDcardNum[0]&(0x80>>i)){
   oddCount++;
  }
 }
 if(evenCount%2 == evenBit && oddCount%2 != oddBit){
  return 1;
 }
 else{
  return 0;
 }
}
void resetData(){
 RFIDcardNum[0] = 0;
 RFIDcardNum[1] = 0;
 RFIDcardNum[2] = 0;
 RFIDcardNum[3] = 0;
 evenBit = 0;
 oddBit = 0;
 recvBitCount = 0;
 isData0Low = 0;
 isData1Low = 0;
 isCardReadOver = 0;
}
// функция interrupt0
void ISRreceiveData0(){
 recvBitCount++;
 isData0Low = 1;
}
// функция interrupt1
void ISRreceiveData1(){
 recvBitCount++;
 isData1Low = 1;
}
```

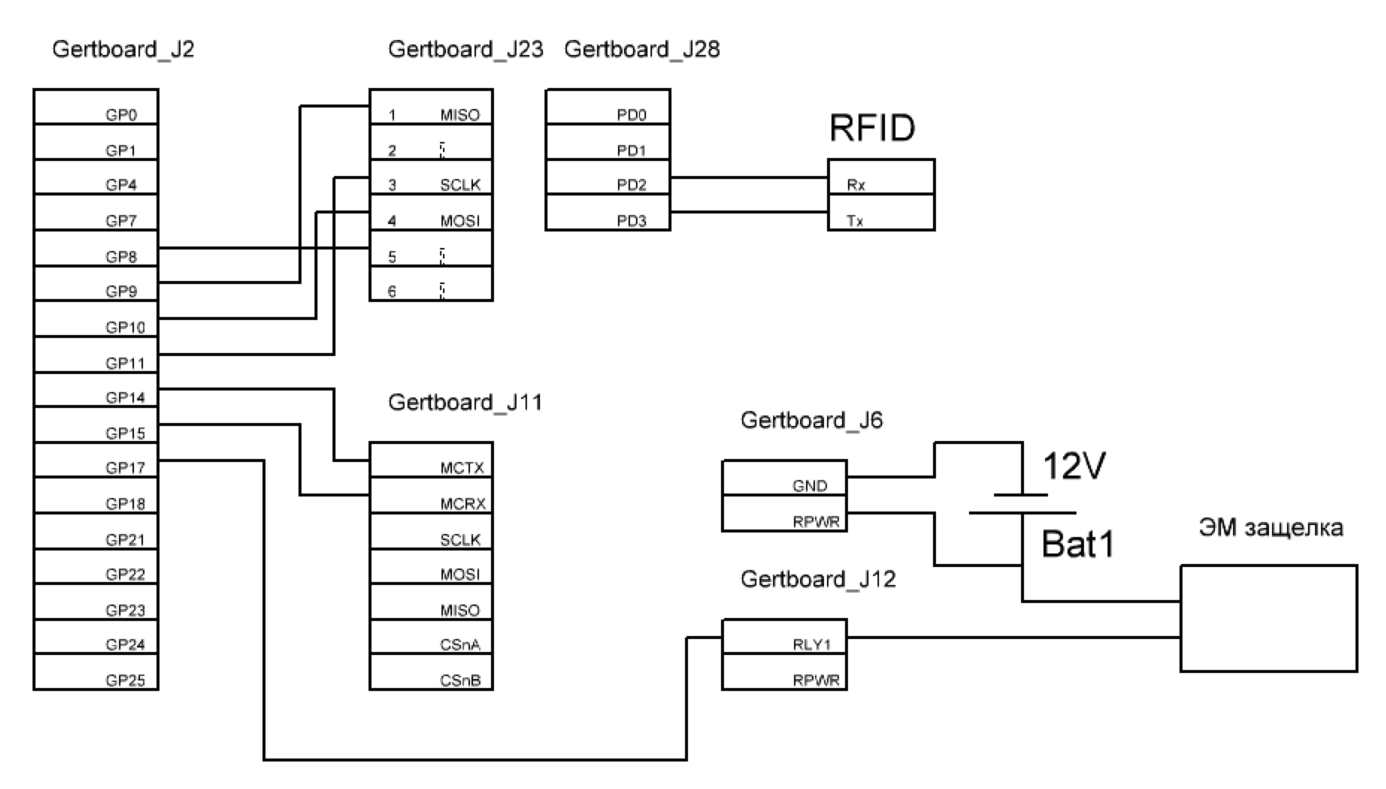


Рис. 6.43. Схема подключения RFID-считывателя

Создание базы данных с использованием Python и MySQL

Теперь необходимо написать для Raspberry Pi скрипт, получающий данные (RFID-код) из ATmega Gertboard и проверяющий наличие кода в базе данных. Для создания базы данных воспользуемся MySQL. MySQL — свободное программное обеспечение, не слишком требовательное к ресурсам. Так что комбинация Python+MySQL — вполне приемлемый вариант для приложений, которые привязаны к обработке данных.

Установим MySQL:

```
sudo apt-get install mysql-server-5.5
```

При установке необходимо ввести новый пароль пользователя root для подключения к MySQL:

```
mysql -u root -p
```

Создадим новую базу данных `my_project1` (рис. 6.44) и в ней две таблицы данных: `base_keys` — для хранения ключей и данных владельцев (ФИО и номер квартиры) и `input_events` — для хранения времени входа, ключа, по которому вошли, и название фото, которое получим с камеры Raspberry Pi Camera Board.

Создание базы данных `my_project1`:

```
CREATE DATABASE my_project1;
```

Создание таблицы `base_keys`:

```
CREATE TABLE `base_keys` (
`ID` bigint(20) unsigned NOT NULL auto_increment,
```

```
`key` varchar(20) NOT NULL default '',
`name` varchar(64) NOT NULL default '',
`houseroom` varchar(7) NOT NULL default '',
PRIMARY KEY  (`ID`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
```

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 46
Server version: 5.5.35-0+wheezy1 (Debian)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE my_project1;
Query OK, 1 row affected (0.00 sec)

mysql> use my_project1;
Database changed
mysql> CREATE TABLE `base_keys` (
    -> `ID` bigint(20) unsigned NOT NULL auto_increment,
    -> `key` varchar(20) NOT NULL default '',
    -> `name` varchar(64) NOT NULL default '',
    -> `houseroom` varchar(7) NOT NULL default '',
    -> PRIMARY KEY  (`ID`)
    -> ) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE `input_events` (
    -> `ID` bigint(20) unsigned NOT NULL auto_increment,
    -> `id_key` bigint(20) NOT NULL default '0',
    -> `photo` varchar(64) NOT NULL default '',
    -> `time_event` datetime NOT NULL default '0000-00-00 00:00:00',
    -> PRIMARY KEY  (`ID`)
    -> ) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.03 sec)

mysql>
```

Рис. 6.44. Создание базы данных и таблиц в MySQL

Создание таблицы `input_events`:

```
CREATE TABLE `input_events` (
`ID` bigint(20) unsigned NOT NULL auto_increment,
`id_key` bigint(20) NOT NULL default '0',
`photo` varchar(64) NOT NULL default '',
`time_event` datetime NOT NULL default '0000-00-00 00:00:00',
PRIMARY KEY  (`ID`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
```

Базу данных необходимо заполнить данными о кодах ключей, выполнив запросы следующего вида:

```
INSERT INTO `my_project1`.`base_keys` (`ID`, `key`, `name`, `houseroom`) VALUES
(NULL, '45678345634', 'Иванов С.П.', '25');
INSERT INTO `my_project1`.`base_keys` (`ID`, `key`, `name`, `houseroom`) VALUES
(NULL, '457567834', 'Петрова А.Г.', '16');
INSERT INTO `my_project1`.`base_keys` (`ID`, `key`, `name`, `houseroom`) VALUES
(NULL, '45678348934', 'Сидоров И.И.', '24');
INSERT INTO `my_project1`.`base_keys` (`ID`, `key`, `name`, `houseroom`) VALUES
(NULL, '4457567834', 'Осипов А.А.', '17');
```

Посмотрим содержимое таблицы (рис. 6.45):

```
SELECT * FROM `my_project1`.`base_keys` LIMIT 0,30;
```

Если база данных заполнена, можно приступать к созданию скрипта на Python для регистрации событий входа.

```
mysql> INSERT INTO `my_project1`.`base_keys` (`ID`, `key`, `name`, `houseroom`)
VALUES (NULL, '45678348934', 'Сидоров И.И.', '24');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO `my_project1`.`base_keys` (`ID`, `key`, `name`, `houseroom`)
VALUES (NULL, '4457567834', 'Осипов А.А.', '17');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM `my_project1`.`base_keys` LIMIT 0,30;
+----+-------------+-----------------------+-----------+
| ID | key         | name                  | houseroom |
+----+-------------+-----------------------+-----------+
|  5 | 45678345634 | Иванов С.П.           | 25        |
|  6 | 457567834   | Петрова А.Г.          | 16        |
|  7 | 45678348934 | Сидоров И.И.          | 24        |
|  8 | 4457567834  | Осипов А.А.           | 17        |
+----+-------------+-----------------------+-----------+
4 rows in set (0.00 sec)

mysql>
```

Рис. 6.45. Просмотр содержимого таблицы `base_keys`

Обработка данных из ATmega Gertboard

Для работы в Python с MySQL необходимо установить модуль `MySQLdb`. Для установки выполняем следующие команды в терминале:

```
sudo apt-get install python-pip
sudo pip install -U pip
sudo apt-get install python-dev libmysqlclient-dev
sudo  pip install MySQL-python
```

Проверить, что модуль установлен успешно, можно запустив Python и задав команду `import MySQLdb`. Если не выдается ошибка, то модуль установлен правильно (рис. 6.46).

```
pi@raspberrypi ~ $ sudo pip install MySQL-python
Downloading/unpacking MySQL-python
  Downloading MySQL-python-1.2.5.zip (108kB): 108kB downloaded
  Running setup.py (path:/tmp/pip_build_root/MySQL-python/setup.py) egg_info for
 package MySQL-python

Installing collected packages: MySQL-python
  Running setup.py install for MySQL-python
    building '_mysql' extension
    gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstrict-pro
totypes -fPIC -Dversion_info=(1,2,5,'final',1) -D__version__=1.2.5 -I/usr/includ
e/mysql -I/usr/include/python2.7 -c _mysql.c -o build/temp.linux-armv6l-2.7/_mys
ql.o -DBIG_JOINS=1 -fno-strict-aliasing -g
    In file included from _mysql.c:44:0:
    /usr/include/mysql/my_config.h:422:0: warning: "HAVE_WCSCOLL" redefined [ena
bled by default]
    /usr/include/python2.7/pyconfig.h:890:0: note: this is the location of the p
revious definition
    gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,-z,relro build/tem
p.linux-armv6l-2.7/_mysql.o -L/usr/lib/arm-linux-gnueabihf -lmysqlclient_r -lpth
read -lz -lm -lrt -ldl -o build/lib.linux-armv6l-2.7/_mysql.so

Successfully installed MySQL-python
Cleaning up...
pi@raspberrypi ~ $ python
Python 2.7.3 (default, Jan 13 2013, 11:20:46)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import MySQLdb
>>>
```

Рис. 6.46. Установка и проверка модуля Python `MySQLdb`

Осталось написать скрипт на Python для получения данных из Gertboard ATmega по последовательному порту. Обратите внимание, что порт этот: /dev/ttyAMA0. При получении данных скрипт проверяет наличие кода в таблице `base_keys` базы данных `my_project1`. Если ключ присутствует, отдается команда на GPIO для открытия двери, с помощью камеры Raspberry Camera Board делается снимок и в таблицу `input_events` вносится соответствующая информация. Содержимое скрипта get_event.py представлено в листинге 6.20.

Листинг 6.20. Скрипт get_event.py

```
#!/usr/bin/python

import serial, time, datetime, os, subprocess
import MySQLdb as db

#initialization and open the port
ser = serial.Serial()
ser.port = "/dev/ttyAMA0"
#ser.port = "/dev/ttyS2"
ser.baudrate = 9600
ser.bytesize = serial.EIGHTBITS
```

```
ser.parity = serial.PARITY_NONE
ser.stopbits = serial.STOPBITS_ONE
#ser.timeout = None          #block read
ser.timeout = 1            #non-block read
#ser.timeout = 2              #timeout block read
ser.xonxoff = False
ser.rtscts = False     #disable hardware (RTS/CTS) flow control
ser.dsrdtr = False       #disable hardware (DSR/DTR) flow control
ser.writeTimeout = 2     #timeout for write

try:
    ser.open()
except Exception, e:
    print "error open serial port: " + str(e)
    exit()

if ser.isOpen():
    try:
      ser.flushInput() #flush input buffer, discarding all its contents
      ser.flushOutput()#flush output buffer, aborting current output
      con = db.connect(host="localhost", user="root", passwd="******",
db="my_project1")
      cur = con.cursor()
      while True:
        response = ser.readline()
        if response=='':
          pass
        else:
           print("read data: " + response)
           cur.execute('SET NAMES `utf8`')
           cur.execute('SELECT `ID` FROM `base_keys` WHERE
`key`='+str(response)+' ')
           print cur.rowcount
           if cur.rowcount>0:
             id_key=0;
             for row in cur:
                id_key=row[0]
             dt = datetime.datetime.now()
             photo=str(time.mktime(dt.timetuple()))+".jpg"
             time_event=datetime.datetime.now().strftime('%Y/%m/%d %H:%M:%S')
             print time_event
             print id_key
             print time.time()
             insertstmt=("insert into input_events (id_key,photo,time_event)
values ('%d','%s','%s')" % (id_key,photo,time_event))
             cur.execute(insertstmt)
             #
             os.system('raspistill -o images/'+photo+' ')
```

```
      ser.close()
    except Exception, e1:
      print "error communicating...: " + str(e1)
else:
    print "cannot open serial port "
```

> **ПРИМЕЧАНИЕ**
>
> Код этого проекта вы найдете в папке *glava_06\input_events* сопровождающего книгу электронного архива (см. *приложение*).

6.4. XMOS StartKIT для Raspberry Pi

Компания XMOS выпустила многоядерный комплект разработки производительностью 500 MIPS для Raspberry Pi и других макетных плат. Плата StartKIT (рис. 6.47) представляет собой платформу разработки ультранизкой стоимости, которая открывает технологию конфигурируемого многоядерного микроконтроллера xCORE для широкого круга пользователей.

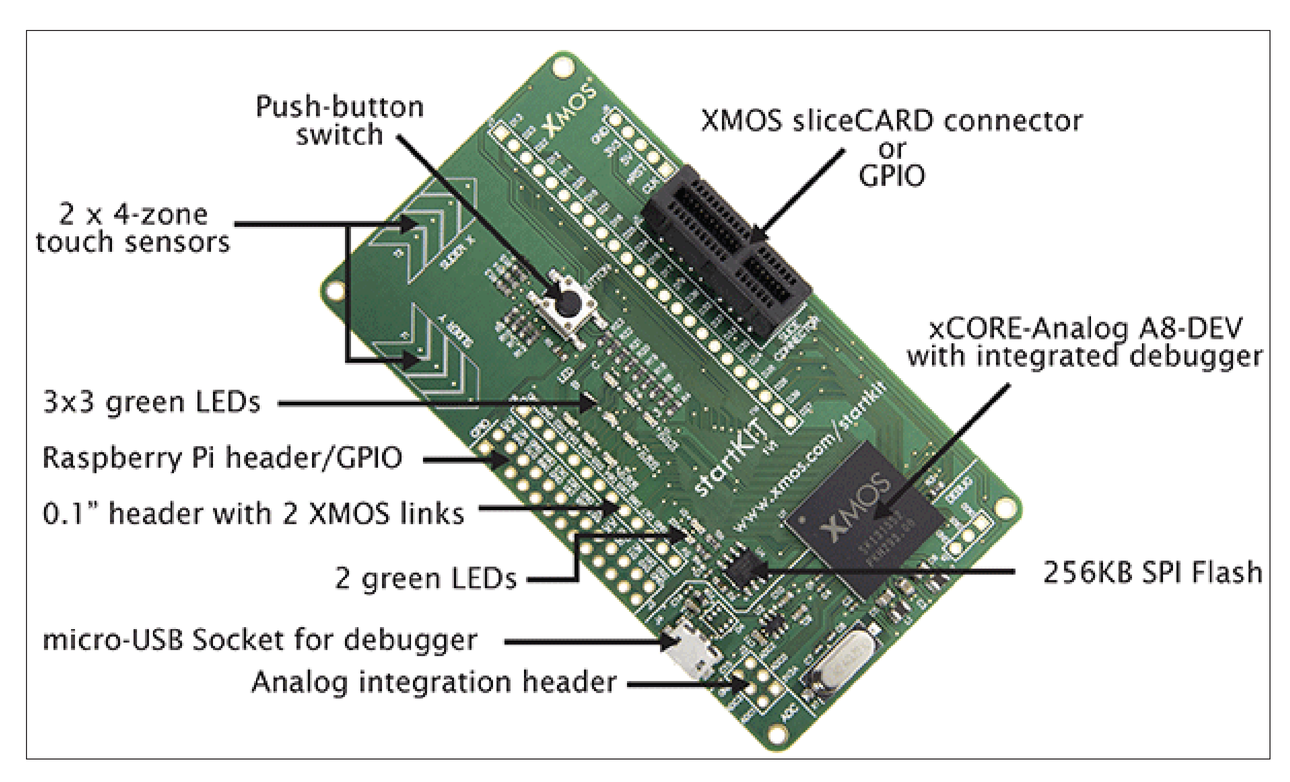


Рис. 6.47. Плата XMOS StartKIT

Плата StartKIT позволяет быстро оценить возможности многоядерных микроконтроллеров xCORE, которые могут быть программно сконфигурированы с большим количеством периферийных и интерфейсных блоков. Набор StartKIT оборудуется разъемами для интерфейсной связи с продуктами Raspberry Pi, делая его идеальным решением для проектов ввода/вывода в реальном режиме времени с использованием платы Raspberry Pi.

Бесплатное средство проектирования xTIMEcomposer позволяет разработчикам удобно и легко запрограммировать требуемую конфигурацию интерфейса и запи-

сать код прикладной программы, используя язык C/C++ в единой комплексной среде программирования. xTIMEcomposer обеспечивает полную графическую поддержку, включая компилятор, отладчик, статический анализатор времени выполнения программы и программное средство логического анализа.

При размерах 94×50 мм плата StartKIT несет на борту аналоговый многоядерный микроконтроллер xCORE XS1-A8-64-DEV производительностью 500 MIPS с 32-битными ядрами для логической обработки. В дополнение к этому StartKIT оснащен массивом светодиодов, нажимным выключателем, двумя датчиками касания и коннектором sliceCARD, который совместим с широким диапазоном портов ввода/вывода, доступных от компании XMOS. Плата также оборудуется разъемом, который позволяет подключить системы на макетных платах.

Встроенный многоядерный микроконтроллер xCORE XS1-A8-64-DEV имеет бортовой отладчик, который разрешает в режиме реального времени проводить полный анализ работы всей схемы. Это позволяет разработчикам видеть, что происходит в интерфейсах устройства и программном коде в режиме реального времени при работающей системе и без ущерба для общей производительности. Аналоговые интерфейсы StartKIT могут контролироваться наряду с цифровыми сигналами — например, пользователи могут контролировать работу емкостных датчиков касания, чтобы видеть сигналы в реальном режиме времени.

Набор стоит 14,99 доллара США. Пользователи, прошедшие регистрацию на сайте производителя до 1 декабря 2013 года, смогли получить одну из 2500 выпущенных плат StartKIT бесплатно. Вот и я уже в конце января 2014 года получил посылку с такой платой.

Для использования StartKIT необходимо скачать среду программирования xTIMEcomposer версии не ниже 13.0.0 со страницы проекта: **https://www.xmos.com/support/downloads/xtimecompose**. Здесь доступны версии среды для операционных систем Windows, AppleMac и Linux (рис. 6.48). Для скачивания необходимо зарегистрироваться.

При первом запуске xTIMEcomposer необходимо ввести свои регистрационные данные (рис. 6.49).

Вводим данные и попадаем в среду программирования xTIMEcomposer studio. Самый быстрый способ начать разработку с использованием StartKIT — импортировать из репозитория xCORE Community приложение GitHub.

Для этого выполняем команду меню **Window | Show View | Developer Column** (рис. 6.50). На вкладке **Developer Column** описаны процессы загрузки с GitHub демонстрационных проектов, их построения и выполнения.

Затем по ссылке **automatic this step** переходим на список демонстрационных проектов (рис. 6.51). Перетаскиваем проект на панель **Project Explorer**, выделяем загруженный проект и для построения проекта выбираем пункт **Project | Build Project**.

Убедившись, что наш набор подключен к компьютеру, переходим по меню **Run | Run Configuration**, в открывшемся окне **Run Configuration** (рис. 6.52) выбираем построенный проект из списка и запускаем его на выполнение кнопкой **Run**.

Как можно видеть, в списке демонстрационных проектов присутствуют проекты **app_spinning_barDefault** — движение зеленых светодиодов (greenLeds) на панели 3×3 и **app_noughts_and_crosses** — он позволяет сыграть с платой в игру "крестики-нолики" на экране 3×3 зеленых светодиодов (greenLeds), используя две панели сенсоров (touch).

Если прогон демонстрационных проектов на StartKIT осуществляется успешно, можно приступать к написанию собственного проекта.

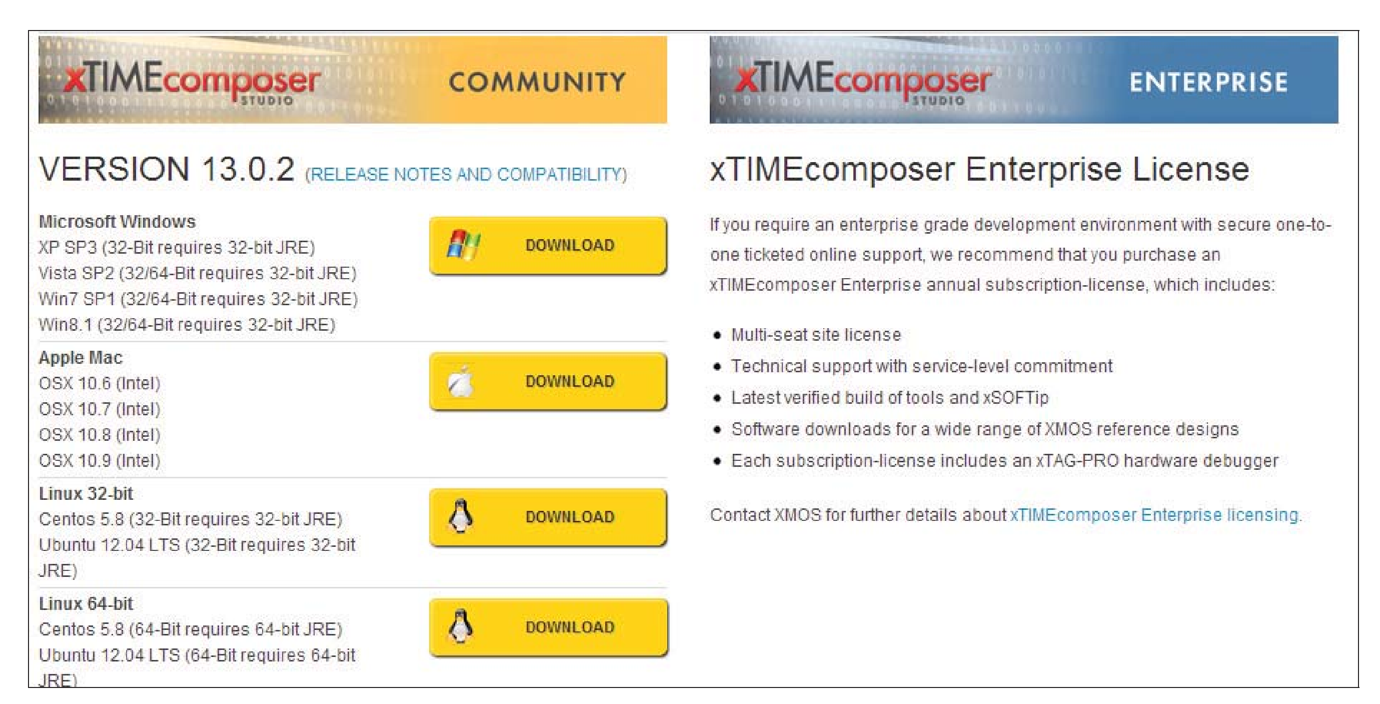


Рис. 6.48. Страница скачивания xTIMEcomposer

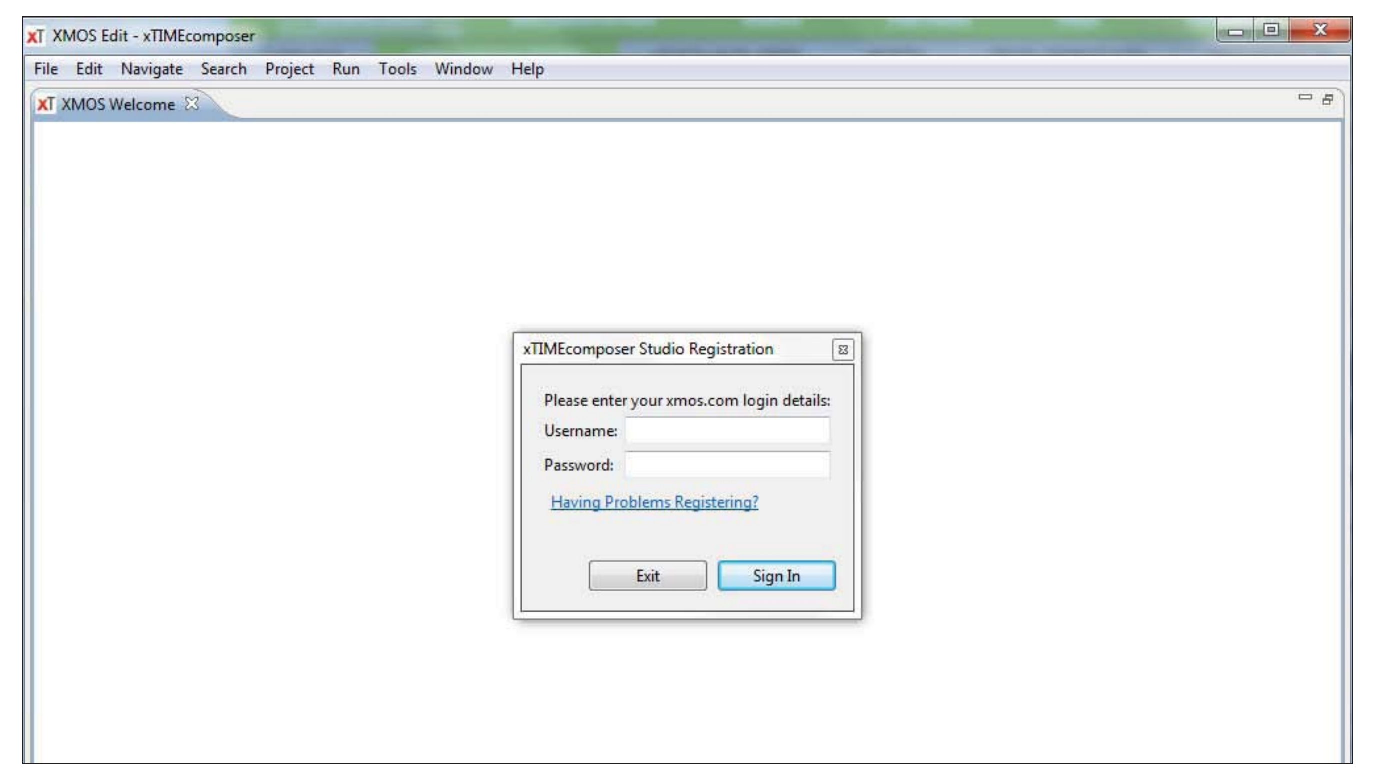


Рис. 6.49. Первый запуск программы

Рис. 6.50. Переход в **Developer Column**

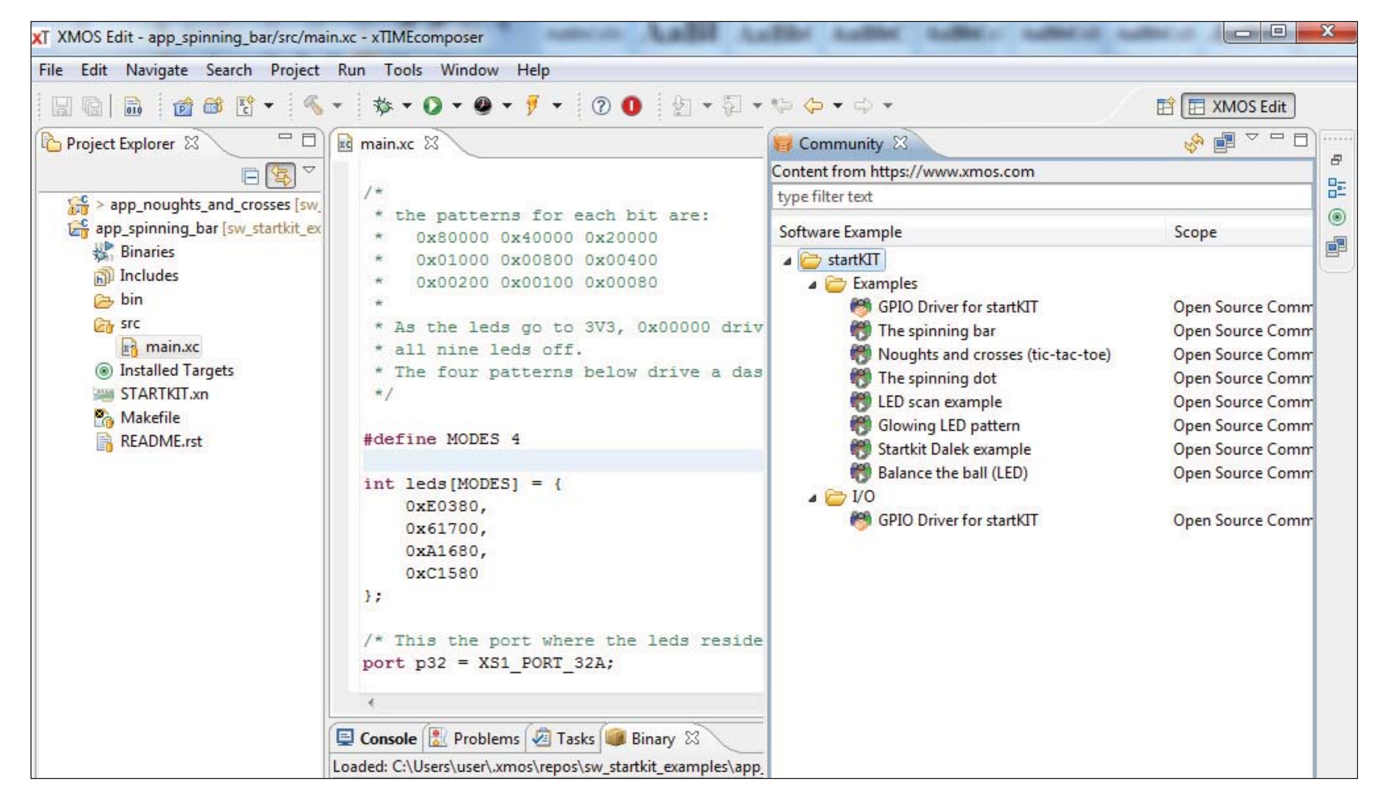


Рис. 6.51. Список демонстрационных проектов для StartKIT

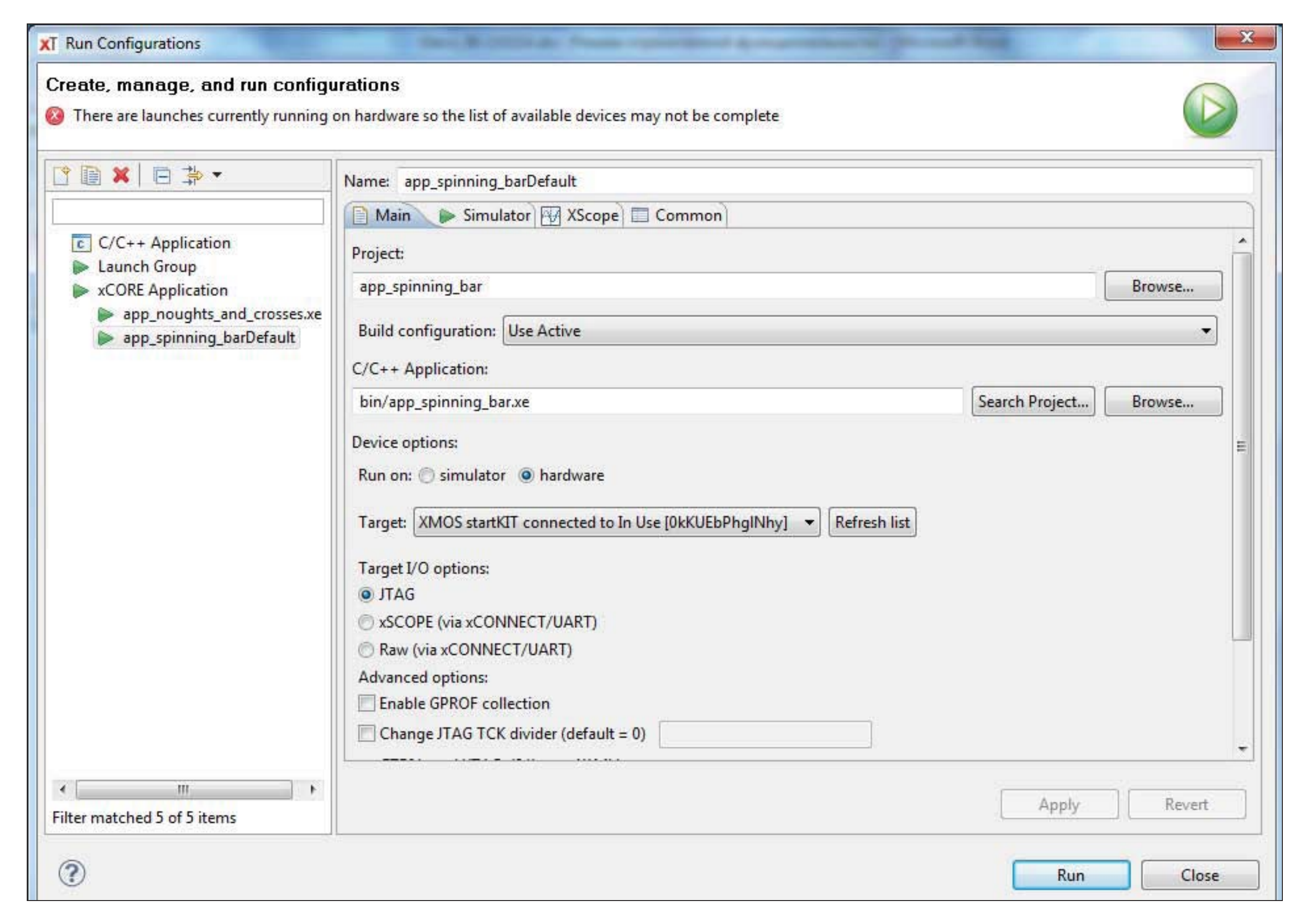


Рис. 6.52. Запуск проектов из панели **Run Configuration**

6.4.1. Подсоединение StartKIT к Raspberry Pi по протоколу SPI

Набор StartKIT оборудуется разъемами для интерфейсной связи с продуктами Raspberry Pi, соответствие разъемов полное, поэтому две платы можно соединить гибким шлейфом (рис. 6.53).

Протокол SPI

SPI — означает Serial Peripheral Interface (последовательный периферийный интерфейс). Интерфейс SPI используется для работы с различными периферийными устройствами — например, с различными ЦАП/АЦП, потенциометрами, датчиками, расширителями портов ввода/вывода (GPIO). С технической точки зрения SPI — это синхронная четырехпроводная шина. Она представляет собой соединение двух синхронных сдвиговых регистров, которые являются центральным элементом любого SPI-устройства. Для соединения используется конфигурация "ведущий-ведомый", и лишь ведущий может генерировать импульсы синхронизации. В схеме всегда только один ведущий (в отличие от той же шины I^2C, где возможен вариант с более чем одним ведущим), а количество ведомых может быть различно. В общем случае выход ведущего соединяется со входом ведомого и наоборот — выход ведомого соединяется со входом ведущего. При подаче импульсов синхронизации на

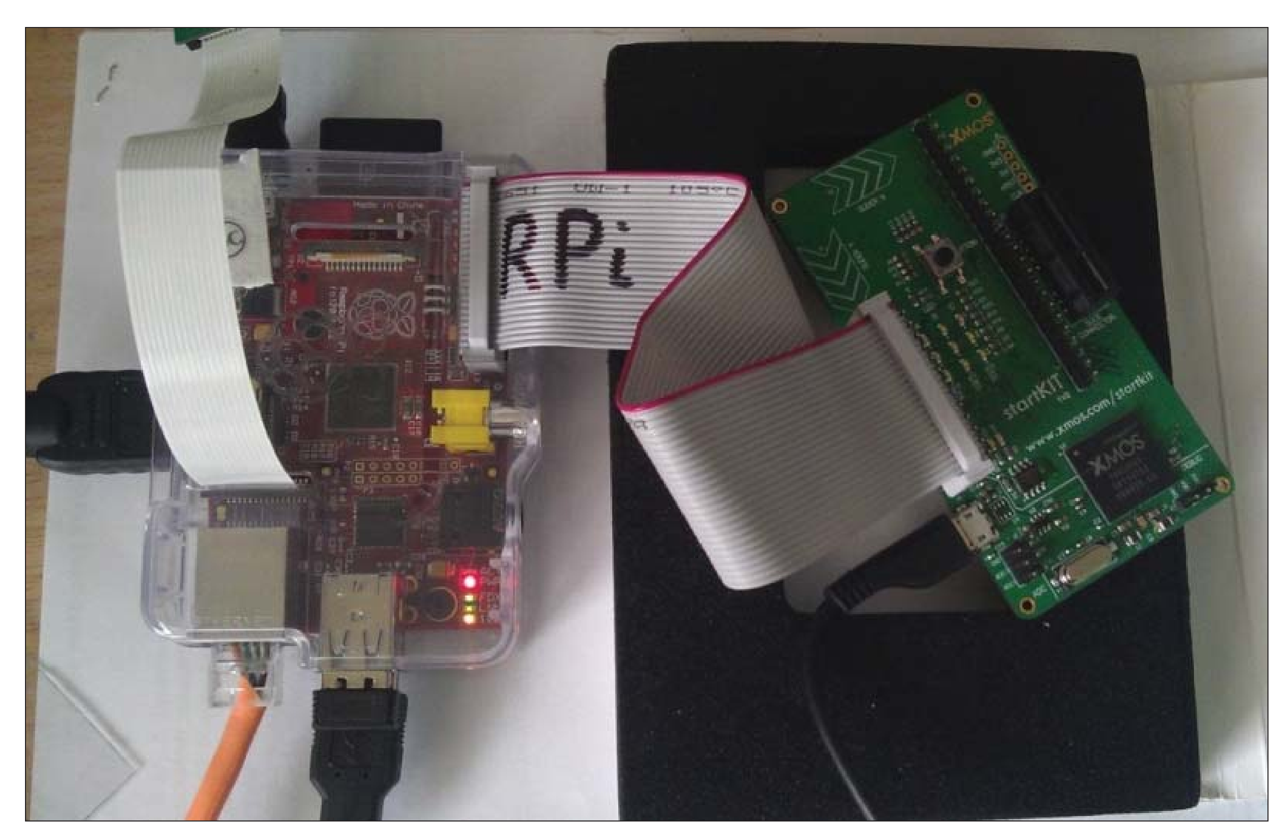


Рис. 6.53. Соединение xMOS StartKIT и Raspberry Pi

выход SCK данные выталкиваются ведущим с выхода MOSI и захватываются ведомым по входу MISO. Таким образом, если подать количество импульсов синхронизации, соответствующее разрядности сдвигового регистра, то данные в регистрах обменяются местами. Отсюда следует, что SPI всегда работает в полнодуплексном режиме.

Контроллер SPI, как правило, реализуется периферийным блоком в микроконтроллере. В большинстве чипов он может работать как в режиме ведущего, так и в режиме ведомого. Но в настоящее время Linux поддерживает только режим ведущего (Master).

Существует несколько способов включения SPI-устройств. Простейший из них вы видите на рис. 6.54.

В данном случае к ведущему все ведомые подключаются параллельно, за исключением сигнала выбора ведомого (SS). Для каждого ведомого необходим отдельный сигнал выбора ведомого (на рисунке они обозначены как SS*x*). В качестве сигналов выбора ведомого могут использоваться как специально предназначенные для этого выходы SPI-контроллера, так и порты ввода/вывода общего назначения (GPIO) микроконтроллера.

Два проводника служат для передачи данных, один для подачи тактовых импульсов и по одному сигналу выбора ведомого для каждого из ведомых.

Описание используемых сигналов:

- MOSI (**M**aster **O**utput, **S**lave **I**nput (выход ведущего, вход ведомого)) — этот сигнал предназначен для последовательной передачи данных от ведущего к ведомому;

- MISO (**M**aster **I**nput, **S**lave **O**utput (вход ведущего, выход ведомого)) — этот сигнал предназначен для последовательной передачи данных от ведомого к ведущему;
- SCLK — **S**erial **Cl**oc**k** (сигнал синхронизации) — используется для синхронизации при передаче данных;
- ~CS — **C**hip **S**elect (выбор микросхемы) — с помощью этого сигнала происходит активация ведомого устройства.

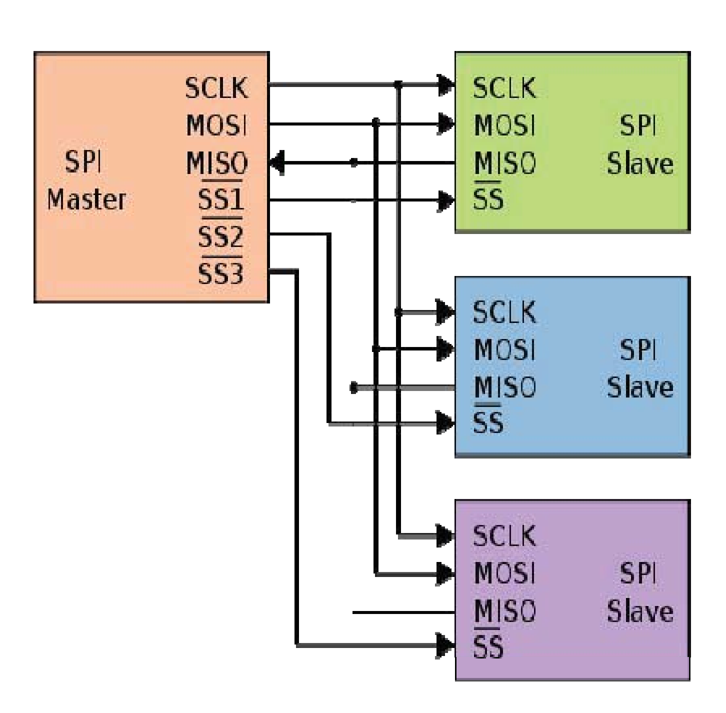


Рис. 6.54. Один из способов подключения SPI

Установка поддержки SPI в Raspberry Pi

Необходимо убедиться, что в Raspberry Pi интерфейс SPI включен (по умолчанию он обычно отключен, потому что редко используется):

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

Закомментируем эту строку (поставив в начале строки #):

```
spi-bcm2708
```

Сохраняем файл (комбинацией клавиш <Ctrl>+<O>) и перезагружаем устройство:

```
sudo reboot
```

После перезагрузки вводим в терминале:

```
lsmod
```

В списке устройств должно присутствовать устройство **spi_bcm2708** (рис. 6.55).

```
pi@raspberrypi ~ $ lsmod
Module                  Size  Used by
lirc_dev               10115  0
snd_usb_audio         116191  0
snd_hwdep               5968  1 snd_usb_audio
snd_usbmidi_lib        18267  1 snd_usb_audio
snd_seq_midi            4682  0
snd_seq_midi_event      6551  1 snd_seq_midi
uvcvideo               69312  0
snd_rawmidi            21357  2 snd_usbmidi_lib,snd_seq_midi
videobuf2_core         30539  1 uvcvideo
videodev              111230  2 uvcvideo,videobuf2_core
media                  13467  2 uvcvideo,videodev
videobuf2_vmalloc       2922  1 uvcvideo
videobuf2_memops        2114  1 videobuf2_vmalloc
joydev                  9084  0
evdev                   9419  3
snd_soc_bcm2708_i2s     5474  0
regmap_mmio             2806  1 snd_soc_bcm2708_i2s
snd_soc_core          131268  1 snd_soc_bcm2708_i2s
regmap_spi              1897  1 snd_soc_core
snd_pcm                81593  2 snd_usb_audio,snd_soc_core
snd_page_alloc          5156  1 snd_pcm
regmap_i2c              1645  1 snd_soc_core
snd_compress            8076  1 snd_soc_core
snd_seq                53769  2 snd_seq_midi_event,snd_seq_midi
snd_timer              20133  2 snd_pcm,snd_seq
snd_seq_device          6473  3 snd_seq,snd_rawmidi,snd_seq_midi
leds_gpio               2059  0
snd                    61291  10 snd_usb_audio,snd_soc_core,snd_hwdep,snd_timer,snd_pcm,snd_seq,snd_
rawmidi,snd_usbmidi_lib,snd_seq_device,snd_compress
led_class               3688  1 leds_gpio
spi_bcm2708             4728  0
8192cu                550816  0
cdc_acm                15827  0
pi@raspberrypi ~ $
```

Рис. 6.55. Проверка подключения поддержки SPI в Raspberry Pi

Модуль *spidev* для Python

Подключим поддержку модуля `spidev` для Python. Модуль нужен для поддержки обмена данных по протоколу SPI.

Приступаем:

```
sudo apt-get update
sudo apt-get upgrade
sudo reboot
```

Установим модуль `python-dev`:

```
sudo apt-get install python-dev
```

Загрузим и установим модуль `spidev`:

```
mkdir python-spi
cd python-spi
wget https://raw.github.com/doceme/py-spidev/master/setup.py
wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
sudo python setup.py install
```

Теперь устройство SPI доступно для программирования на Python.

Пишем скрипт на Python для обмена данными по протоколу SPI. Содержимое скрипта представлено в листинге 6.21.

Листинг 6.21. Скрипт simple-spi.py на Python для обмена данными по протоколу SPI

```
import spidev
import time

spi = spidev.SpiDev()
spi.open(0,0)
spi.mode = 3
spi.max_speed_hz = 4000000

count = 0
while True:
    tx = [1, 2, 3, 4 ,5 ,6,7, count]
    print "Sending " + str(tx)
    resp = spi.xfer2(tx)
    print "Received" + str(resp)
    count += 1
    if count > 255: count = 0
    time.sleep(1)
```

Запускаем скрипт на выполнение:

```
python /home/pi/python_prg/startkit/simple-spi.py
```

И видим в терминале процесс отправки данных по SPI (рис. 6.56).

```
pi@raspberrypi ~ $ python /home/pi/python_prg/startkit/simple-spi.py
Sending [1, 2, 3, 4, 5, 6, 7, 0]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 1]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 2]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 3]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 4]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 5]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 6]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 7]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 8]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 9]
Received[0, 0, 0, 0, 0, 0, 0, 0]
Sending [1, 2, 3, 4, 5, 6, 7, 10]
Received[0, 0, 0, 0, 0, 0, 0, 0]
```

Рис. 6.56. Обмен данными по протоколу SPI

6.4.2. Создание программы для XMOS StartKIT

На сайте **http://www.xcore.com/projects/raspberry-spi** выложен проект для обмена данных StartKIT с Raspberry Pi по протоколу SPI (рис. 6.57). Скачиваем этот проект и распаковываем. Затем импортируем его в xTIMEcomposer studio: **File | Import | Existing Projects into Workspace**. Указываем путь к папке с проектом (рис. 6.58), нажимаем кнопку **OK** — проект появляется в панели **Project Explorer** (рис. 6.59).

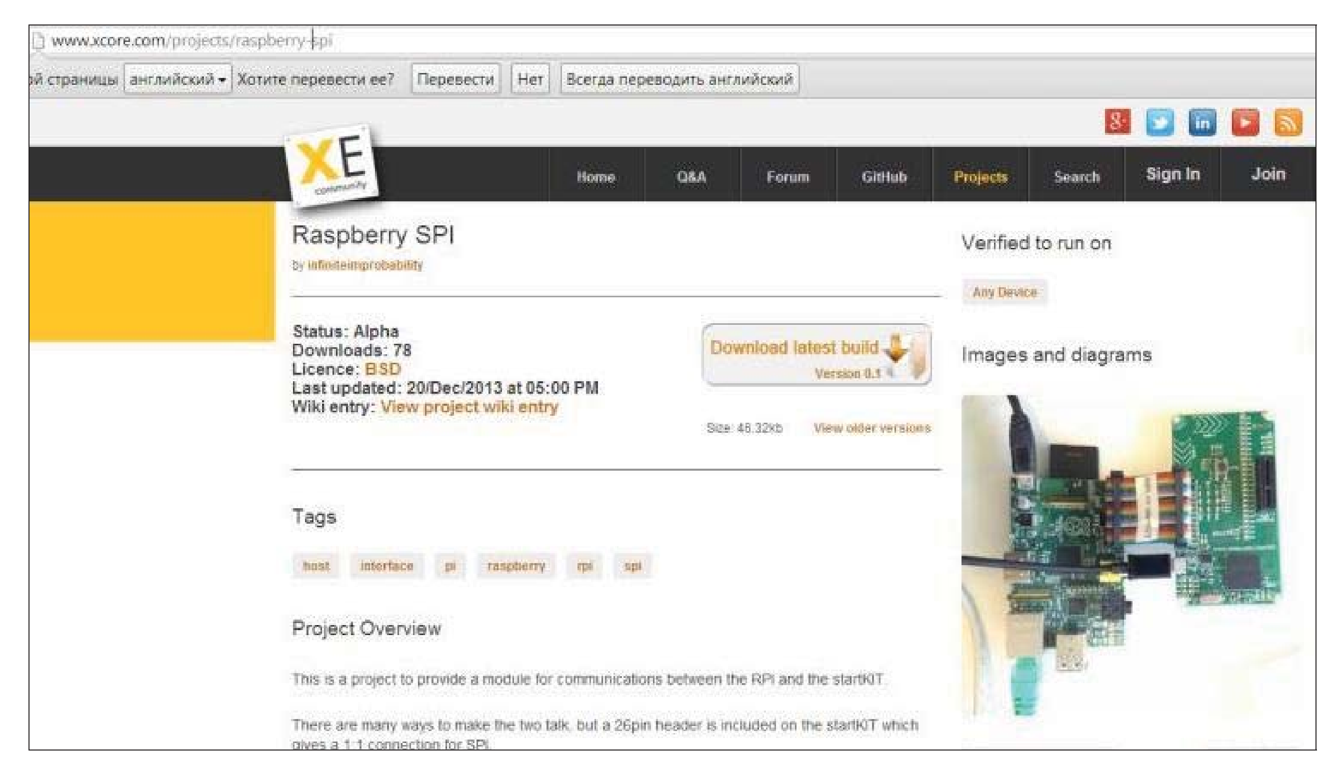


Рис. 6.57. Страница проекта **Raspberry SPI**

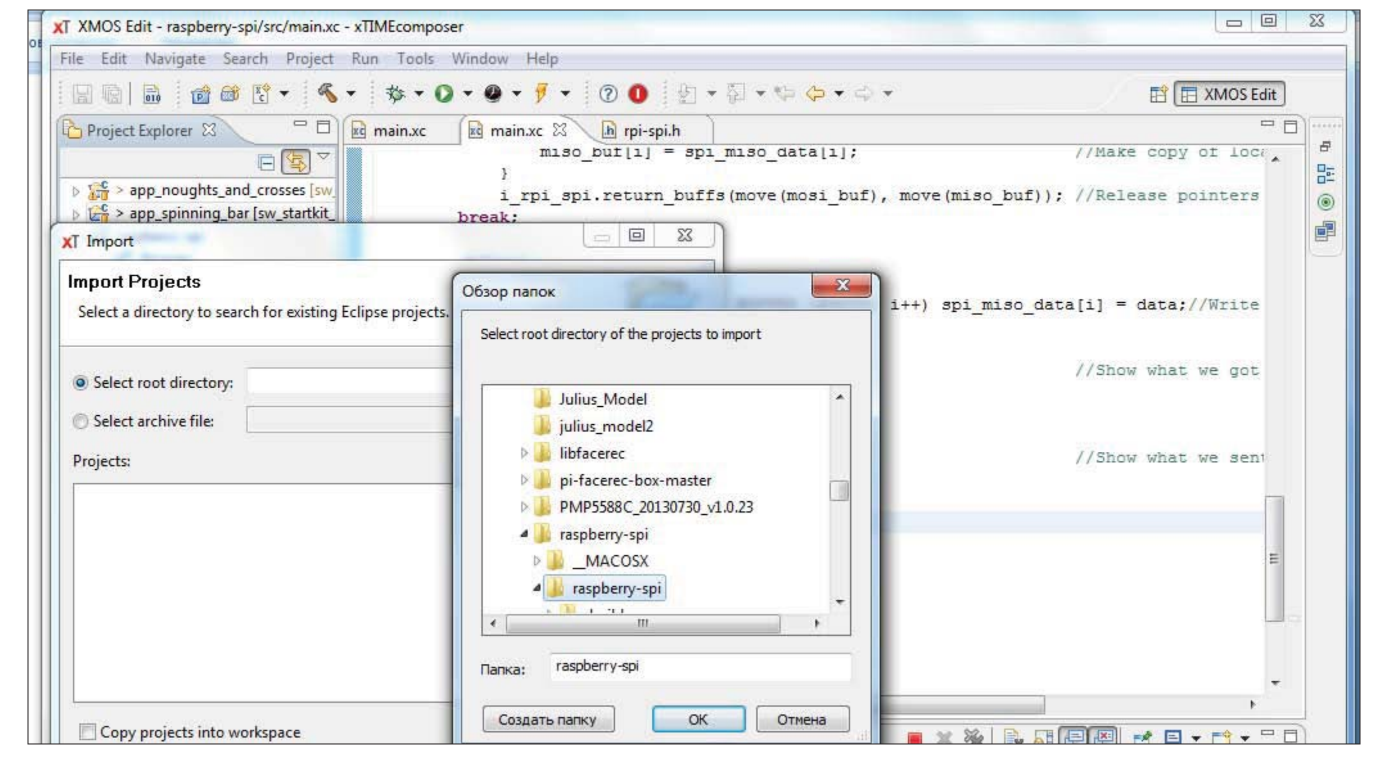


Рис. 6.58. Импорт проекта в xTIMEcomposer studio

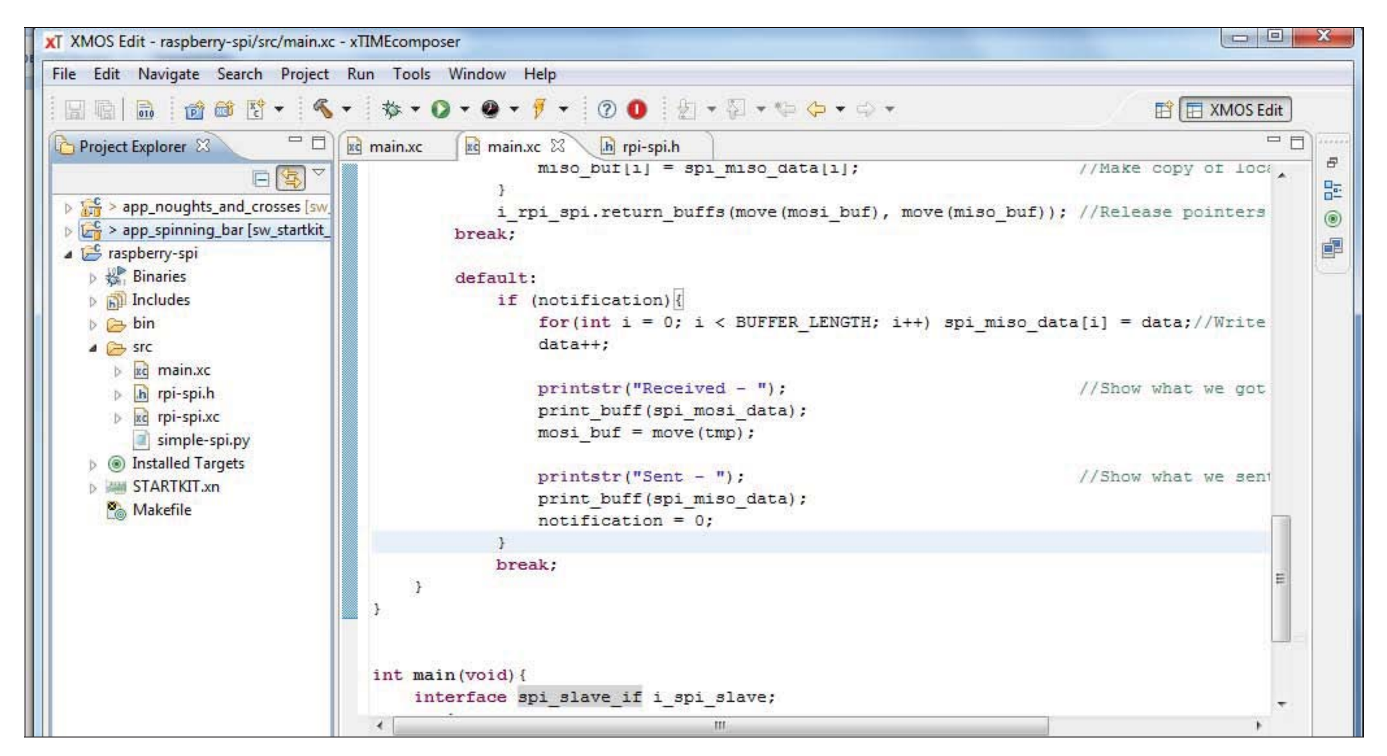


Рис. 6.59. Проект **Raspberry SPI** импортирован в xTIMEcomposer studio

Выставляем для проекта нужные настройки (рис. 6.60) и запускаем его через меню **Run configuration**.

Затем на Raspberry Pi запускаем скрипт simple-spi.py (см. *разд. 6.4.1*, листинг 6.21) и видим обмен данными между двумя устройствами (рис. 6.61).

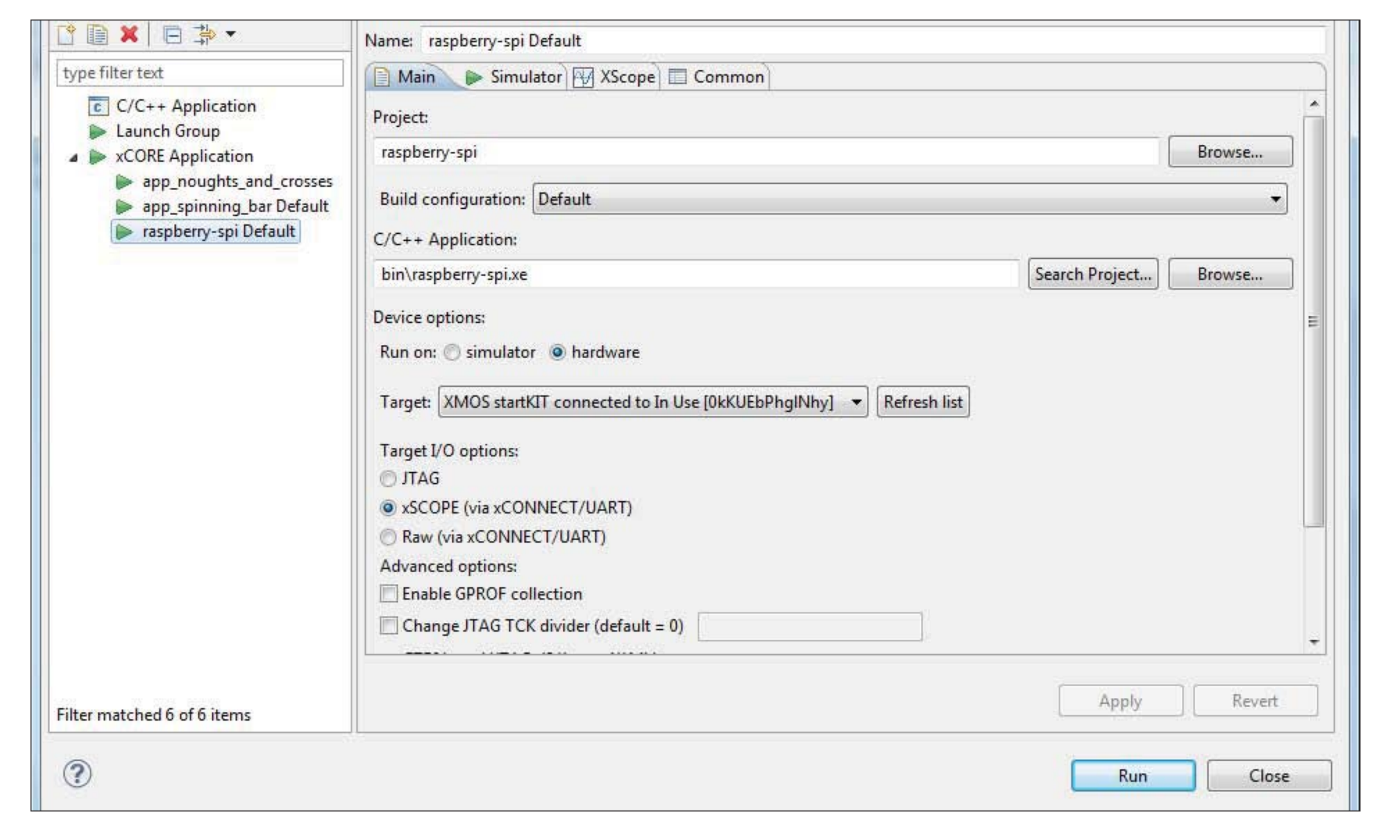


Рис. 6.60. Настройки в xTIMEcomposer studio для запуска

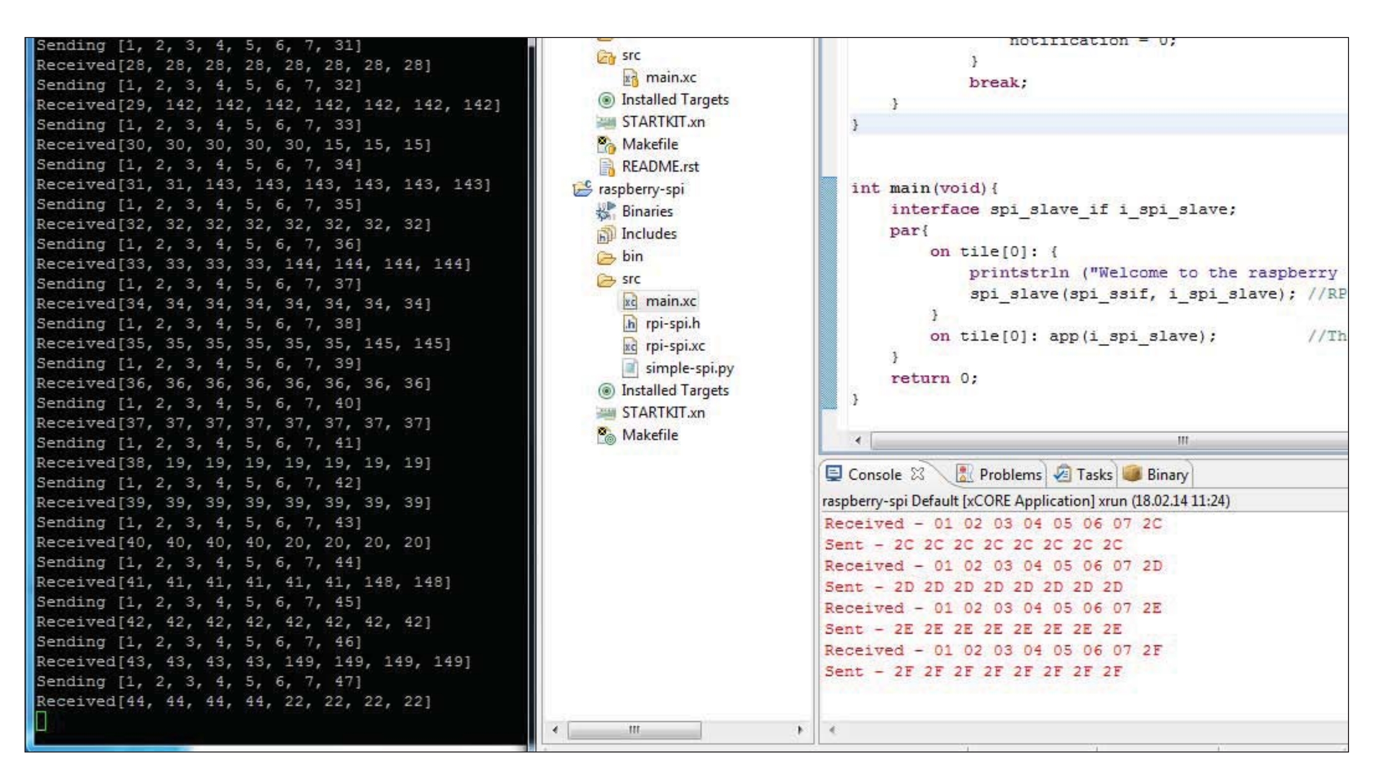


Рис. 6.61. Обмен данными по SPI StartKIT и Raspberry Pi

Примечание

Коды этого проекта вы найдете в папке *glava_06\raspberry-spi* сопровождающего книгу электронного архива (см. *приложение*).

На сайте **http://www.xcore.com/projects** имеются и другие примеры использования платы StartKIT.

Заключение

Выпущенный в свет в 2012 году крошечный компьютер Raspberry Pi снискал огромную популярность и разошелся по миру в количестве более 2,5 миллионов штук. Основными причинами такой популярности явились его цена ($25 за модель "A" или $35 за модель "B"), доступность, расширяемость и развивающееся сообщество поддержки.

В этой книге мы познакомились с устройством микрокомпьютера Raspberry Pi, основными операционными системами для него, а дистрибутивы Raspbian и Raspbmc изучили подробно. Здесь представлены практические проекты применения Raspberry Pi в различных областях: как сервер видеонаблюдения, для распознавания речи, в "компьютерном зрении", в качестве домашней метеостанции, домашнего медиацентра и пр.

Одним из важнейших применений Raspberry Pi, благодаря наличию у него портов GPIO, является использование компьютера в проектах электроники и робототехники, в связи с чем мы рассмотрели также основные интерфейсные платы, сопрягаемые с Raspberry Pi, и методы программирования таких связок.

Буду рад, если мой материал окажется полезен вам при создании собственных проектов.

ПРИЛОЖЕНИЕ

Описание электронного архива

Электронный архив с материалами, сопровождающими книгу, можно скачать с FTP-сервера издательства по ссылке **ftp://ftp.bhv.ru/9785977535199.zip**, а также со страницы книги на сайте **www.bhv.ru**.

В архиве находятся следующие папки:

- \glava_04 — исходники примеров и проектов *главы 4*;
- \glava_05 — исходники примеров и проектов *главы 5*;
- \glava_06 — исходники примеров и проектов *главы 6*;
- \cxem — электрические схемы в формате spl7.